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
－Eclipse プラグインの GUI 設計、開発－

成澤 翔平

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

本報告書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特定課題研究「スケールアウト型データベース製品に関わる設計開発」テーマに関する報告書である。

本報告書で述べるプロジェクトはスケールアウト型データベース製品である InfoFrame Relational Store（以下、IRS）に対応したデータ移行支援ツールを開発する。これにより、ビッグデータ分析におけるユーザーの知識的、技術的負担を軽減することが主たる目的である。

筆者は、主にデータ移行支援ツールの開発において GUI を担当し、提案、設計、実装、テストまで行った。GUI の設計において将来的な拡張を考慮し、MVC パターンを適用した設計を積極的に取り入れた。また、ユーザーの習熟コストを削減するために、既存ツールや Eclipse の操作特性を取り入れて開発を行った。また、将来的な開発を想定して担当毎に拡張性向上の創意工夫を行った。

プロジェクト全体としては IRS に対応したデータ移行支援ツールの基本的な機能の開発を終え、本プロジェクトの顧客である日本電気株式会社に納品を行った。しかし、開発の遅延が発生したため機能の一部の完成を見送る形となった。

まず、本プロジェクトの開発背景や提案システム、プロジェクト体制などについて本報告書では述べる。次に本プロジェクトで筆者が担当した GUI 機能の開発、既存ツールや Eclipse の調査、テストなどについて報告するものである。

目次

図目次

第1章　はじめに

本章では問題背景、現状の課題を述べ、それを踏まえてプロジェクトの概要、目的及び担当部分の概要について述べる。最後に本報告書の構成について述べる。

1.1　問題背景

ビッグデータを Business Intelligence ツール（以下、BI ツール）を用いて分析し、企業活動に取り入れようとする動きが活発になっている[1][2]。その際、ビッグデータを蓄積しているデータベースから BI ツールへのデータ移行がユーザーにとって大きな技術的、知識的負担となっている[3]。そのため、Extract-Transform-Load[4](以下、ETL)ツールを用いて、データ移行に係るユーザーへの負担を大きく軽減する場合が多い。

ETL ツールは DB や CSV など様々なデータソースからデータを抽出し、適切な形に変換して、BI ツールが持つ DB や Data Warehouse（以下、DWH）に格納するツールである[5]。ユーザーは BI ツールにデータを格納する際に DB やプログラミングの専門的な知識なしに、データソースからデータを抽出し、適切な形に変換して、格納することが可能である。

1.2　データベースの問題

ビッグデータ分析が活発になっているが、一方で DB へのビッグデータの蓄積が問題になっている。一般的に利用されているリレーショナルデータベース(以下、RDB)は標準化されており、長年利用されているため信頼性も高い。しかし、ビッグデータ分析を行う上では拡張性に問題があり、今後増え続けるデータに対応することは難しい。これに対して、Key-Value Store(以下、KVS)などの No SQL 型の DB はスケールアウト性が高く、大量のデータに対応することが可能である。しかし、RDB とは違い標準化されていないため、製品独自 API への対応や専門の技術者などが必要になり、安易に導入することはできない[6]。

1.3　InfoFrame Relational Store

1.2 節で述べたように、ビッグデータ分析を行う上で DB にはいくつかの問題があった。そこで、日本電気株式会社(以下、NEC)は RDB の汎用性と KVS のスケールアウト特性を兼ね備えた DB ソフトウェアの InfoFrame Relational Store[7] (以下、IRS)を開発した。これにより、RDB の汎用的な操作で KVS の高いスケールアウト特性を発揮することを期待されている。

1.4　IRS を利用したビッグデータ分析環境の課題

現在 IRS に対応した ETL ツールは存在しない。そのため、IRS を用いたビッグデータ分析環境(図 1.1)では IRS からデータを移行する際、ユーザーに対して多くの専門的な知識や技術を要求する。要求される知識や技術、問題について以下に示す。

1. **IRS や移行先 DB などのデータソースに対する専門的な仕様や知識**
   データを移行するためには様々なデータソースの仕様を把握し、それに合わせた独自の方法でデータを抽出する必要がある。また、データソースによってデータの型が違うた

め、その対応付けも行う必要があり、多くの知識が要求される。

2. **データ移行、格納のためのビッグデータプログラミング技術**
   ビッグデータ分析では通常のプログラミングとは違い、巨大なデータを扱うため特殊なプログラミングが必要となる。また、パフォーマンスを出すためには高度な技術と経験が必要となり、容易にプログラミングを行うことはできない。
3. **作成したプログラムの維持、管理**
   専門的な知識と高度な技術を持つ技術者によって作られたプログラムを維持、管理することは難しい。さらに、データ移行の度にプログラムが増えるため、管理が煩雑になりやすく、担当の技術者が変わると活用できなくなる可能性も十分にある。

よって、ETL ツールなしでデータを移行するためには専門的な知識と高度な技術を持った専任の技術者を用意する必要があり、大きなコストが発生していると予想される。

したがって、IRS に対応した ETL ツールが存在しないことが大きな課題であり、これを開発することで上記の問題を解決、もしくは要求される水準を下げることができると考えられる。

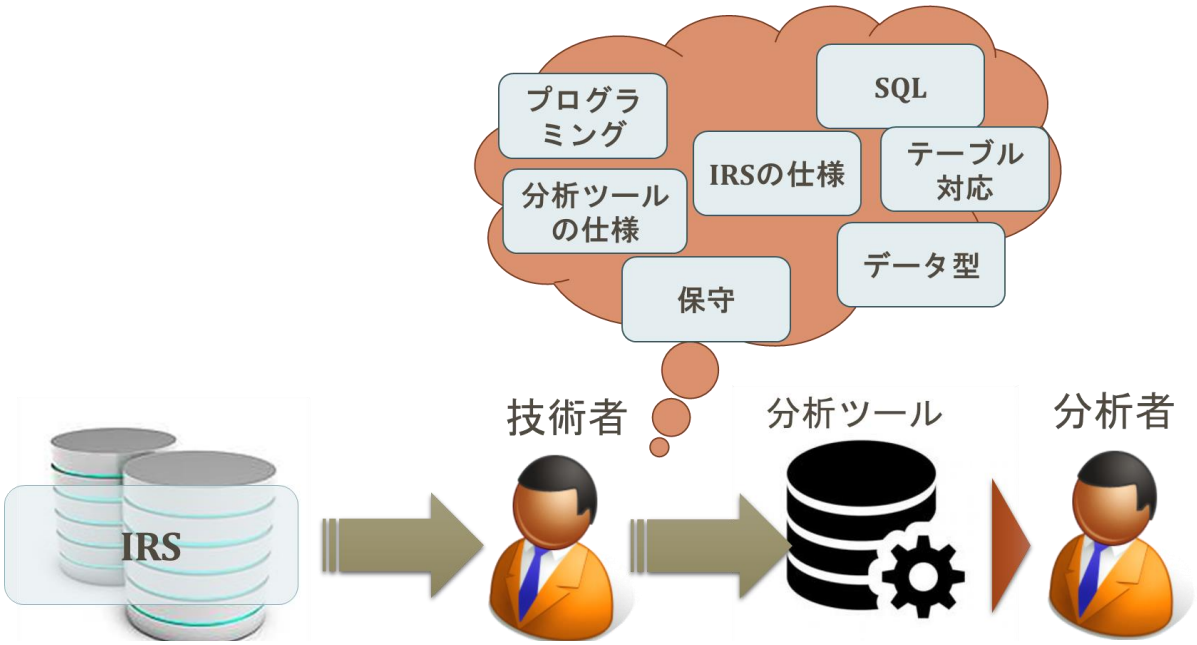


図 1.1 現状の IRS を用いたビッグデータ分析環境

1.5 プロジェクトの概要

本プロジェクトは 2014 年度の研究開発プロジェクトである。そのため、筑波大学から斎藤、HU、唐及び筆者の成澤の四人でチーム開発を行う。また、顧客として NEC に、課題担当教員として筑波大学から天笠俊之准教授に参加していただいく。そして、NEC の要求に基づき IRS に対応した ETL ツール（以下、NET ツール）の開発を行う。

また、本プロジェクトの目的は NET ツールを開発することで、要求される知識や技術の水準を下げ、ユーザーへの負担を軽減することである。

1.6　担当部分の概要

本プロジェクトにおいて筆者は主に NET ツールの GUI 開発を担当した。担当者として筆者は開発において顧客への GUI の提案、GUI の内部設計、実装、テストなどを行った。また、NEC に提案を行うために既存 ETL ツールの調査を行い、開発を行うために Eclipse の調査などを行い、チーム全体での共有を行った。

1.7　本報告書の構成

本報告書では、初めにプロジェクト体制やステークホルダーなどのプロジェクト全体について述べる。次に、本プロジェクトにおいて主に筆者が担当した箇所について述べる。筆者が主に担当したのは ETL ツールと Eclipse の調査や GUI の設計、実装及び担当箇所のテストである。そして、最後に振り返りを行い、反省点や今度の課題について述べる。

第2章　　プロジェクト概要

本章では本プロジェクト体制や顧客の要求、それを受けての我々の提案などプロジェクト全体について詳細を述べる。

2.1　　プロジェクト体制

表 2.1 に本プロジェクトの体制表を示す。本プロジェクトでは、斎藤、HU、唐及び成澤の四人で開発を行う。顧客として NEC に参加していただき、IRS の提供や顧客として開発システムへの要求を行っていただく。また、課題担当教員として筑波大学から天笠俊之准教授にプロジェクトへ参加していただく。

表 2.1 プロジェクト体制表

顧客		NEC
筑波大学	担当教員	天笠 俊之 准教授
	学生	斎藤　優太
		成澤 翔平
		HU BEIQI
		唐 可

2.2　　顧客の要求

顧客の要求は IRS に蓄積されたデータを他のデータベース製品へ移行するツールの設計、開発である。さらに、「IRS の Hadoop API に対応していること」や「GUI によって IRS,DWH 間の対応関係の定義をする」機能も要求されている。

特に「GUI によって IRS,DWH 間の対応関係の定義をする」機能についてはイメージ図として図 2.1 が顧客から提案されている。顧客の提案ではデータ移行における移行元と移行先の対応関係が線を結ぶことで定義されている。そのため、図 2.1 では移行元のテーブルとして転送元 DB のテーブル「表 2」と、移行先のテーブルとして転送先 DB のテーブル「表 a」が線で結ばれており、直感的にデータの流れを GUI で確認することができる。

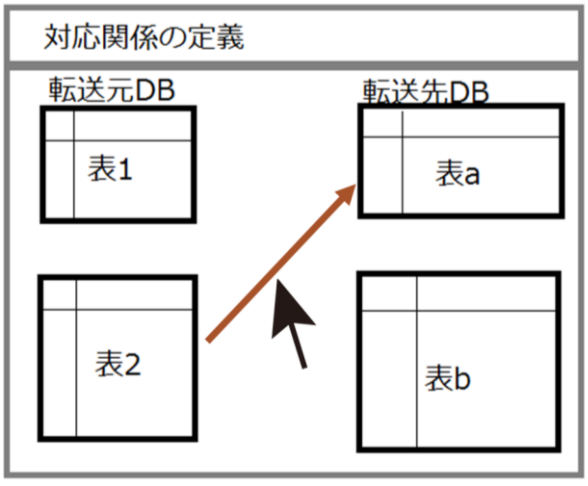


図 2.1 顧客提案イメージ図

2.3　　システム提案

本プロジェクトでは2.2節の要求を踏まえて、IRSに対応したETLツールの提案を行った。提案したシステム図を図 2.2 に示す。NET ツールは IRS からデータを抽出し、適切な形に変換、指定の分析ツールに格納することが可能である。さらに、ユーザーは GUI からそれらの機能を利用することができ、専門的な知識なしに分析ツールへのデータの移行が可能となる。

また、NET ツールの主な特徴は以下の通りである。

- IRS の Hadoop API を利用してデータ抽出が可能である
- プログラミングではなく、GUI でデータ移行の対応付け操作が可能である
- NET ツールのみでデータの抽出、変換、挿入が可能である

そのため、データ移行を行っていた専任の技術者の負担が軽減されることが期待される。また、技術者ではなく分析者であっても NET ツールを用いることでデータ移行が可能になると考えられる。

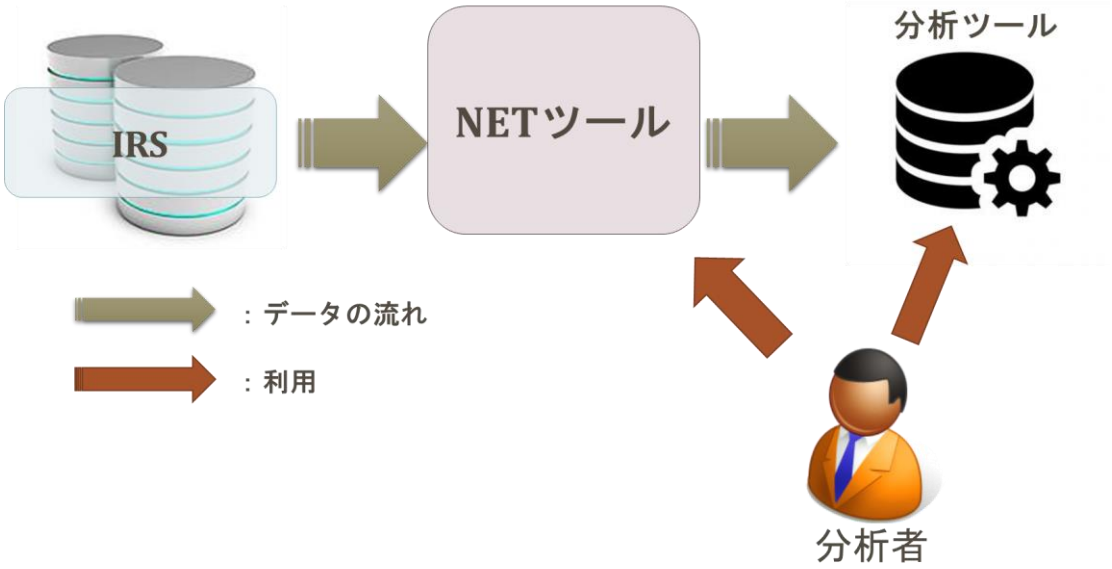


図 2.2 提案システム図

2.4　　提案システムの構成

本節では 2.3 節で提案した NET ツールのシステム構成について述べる。本プロジェクトにおいて開発する NET ツールは抽出(Extract)、変換(Transform)、格納(Load)及び GUI の四つで構成される。NET ツールにおける、それぞれの概要を以下の表 2.2 に示す。

表 2.2 提案システム構成概要

機能名	概要
Extract	Extract機能とはデータソースからデータの抽出を行う機能である。NETツールではIRSのHadoopAPIを利用して、IRSから任意のデータを抽出する。
Transform	Transform機能とは抽出したデータを任意に変換する機能である。NETツールでは抽出したデータに対して四則計算やデータの型変換などを行う。
Load	Load機能とは抽出、変換したデータを任意のデータベースに格納する機能である。NETツールではPostgreSQLに対してデータの格納を行う。
GUI	GUIとはユーザが視覚的にコンピュータを扱うためのインターフェースである。NETツールではEclipseベースでGUIを開発し、上記の機能をサポートする。

2.5 開発機能の役割分担

2.4 節の提案システムの構成から本プロジェクトにおける開発機能の役割分担を行った。著者は主に GUI の設計、開発、テストを担当した。しかし、本プロジェクトの開発期間を考慮し、Transform 機能に係る GUI については同プロジェクトメンバーの HU BEIQI が担当することとなった。以下にメンバー全員の主な役割を表 2.3 で示す。

表 2.3 開発機能担当表

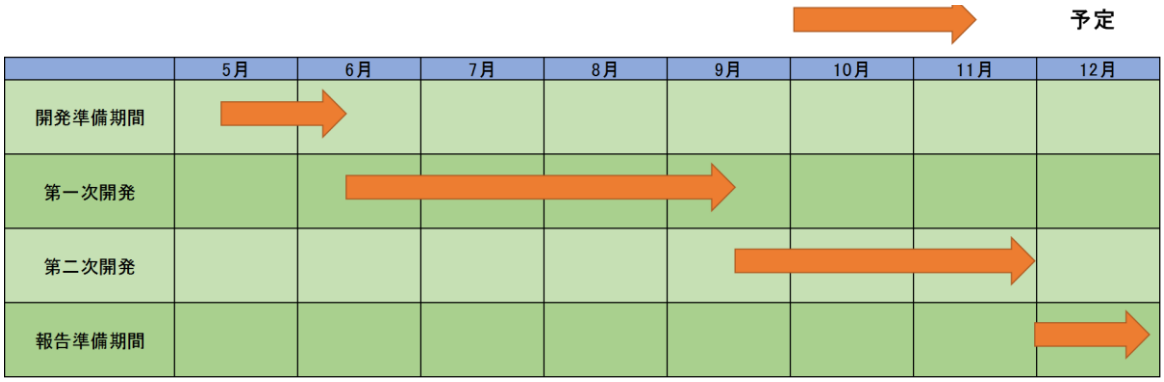
	担当表
唐 可	Extract 機能の設計、実装、テスト Hadoop 環境の構築や障害対処
HU BEIQI	Transform 機能の設計、実装、テスト NET ツールの総合テストやユーザーマニュアルの作成
斎藤 優太	Load 機能の設計、実装、テスト NET ツールの性能調査やセットアップマニュアルの作成
成澤 翔平	ほぼ全ての GUI の設計、実装、テスト ユーザーマニュアルの作成

2.6 開発スケジュール

本節では 2.3 節、2.4 節で述べた提案システムの開発スケジュールを表 2.4 に示す。表 2.4 の通りに開発スケジュールは開発準備期間、第一次開発、第二次開発、報告準備期間の四つの期間に分けられる。

まず、開発準備期間は開発方針や全体の開発スケジュールについて顧客と確認する期間である。次に、第一次開発は基本的な機能を実装する期間である。基本的な機能には、データの抽出、変換、格納やそれらを操作するために必要な GUI などが含まれる。第二次開発では基本的な機能の開発の結果を受けて、顧客からレビューをいただいて優先度の高い機能を追加開発する期間である。最後に、報告準備期間は特定課題研究報告書の作成を行うための期間である。

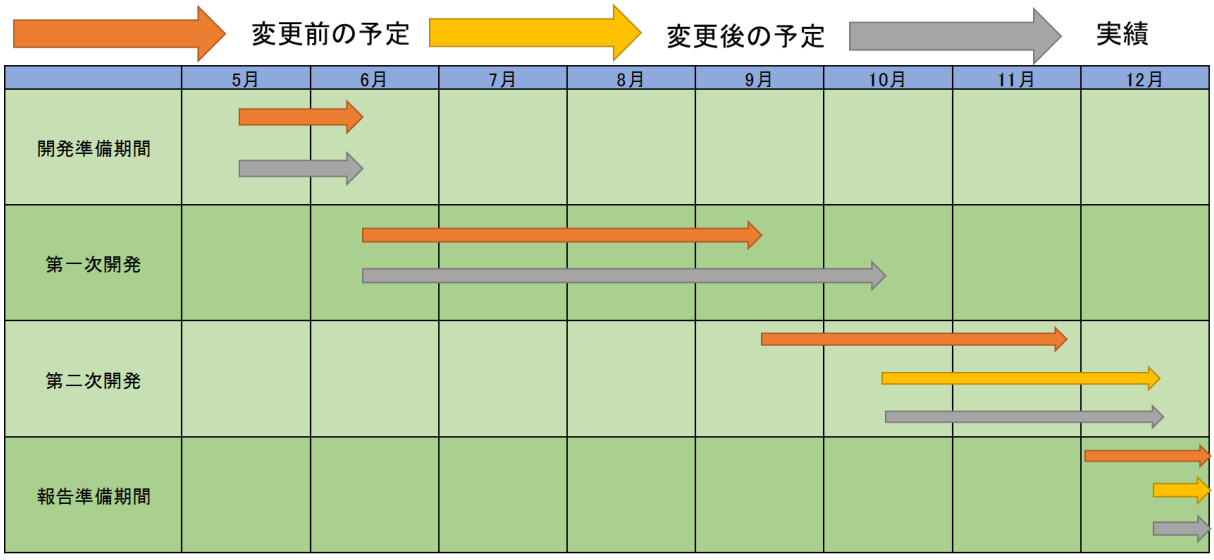
表 2.4 開発スケジュール



2.7 開発スケジュール実績

まず、本プロジェクトでは開発スケジュール全体を 9 月 19 日に変更した。理由としては第一次開発で遅れが発生したためである。全体スケジュールの変更前の予定、変更後の予定及び実績を表 2.5 に示す。また、表 2.5 では変更前の予定を橙色、変更後の予定を黄色、実績を灰色として示す。

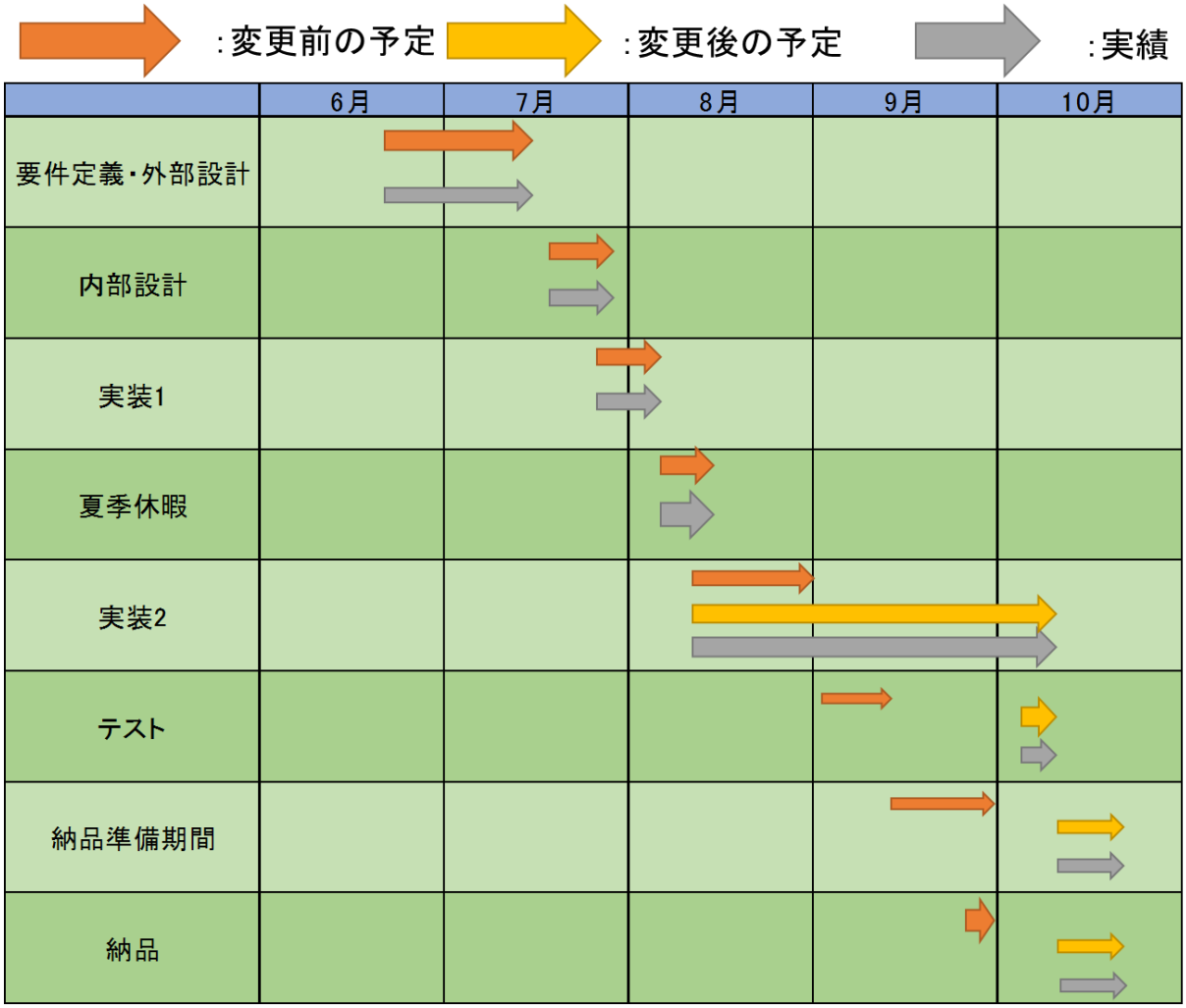
表 2.5 開発スケジュール実績



2.7.1 第一次開発：基本的機能の開発

第一次開発のスケジュールと実績を表 2.6 に示す。表 2.6 から見て取れるように第一次開発の実装２工程において大きな遅延が発生した。これにより、2.7 節で述べたようにスケジュールを変更した。遅延の主な原因は見積もり能力の不足と消極的なソースコード結合による実装期間の延長であった。また、遅延の発見が遅れたことからプロジェクト全体として積極的に進捗を管理する仕組みの不足も挙げられた。

表 2.6 第一次開発スケジュール実績



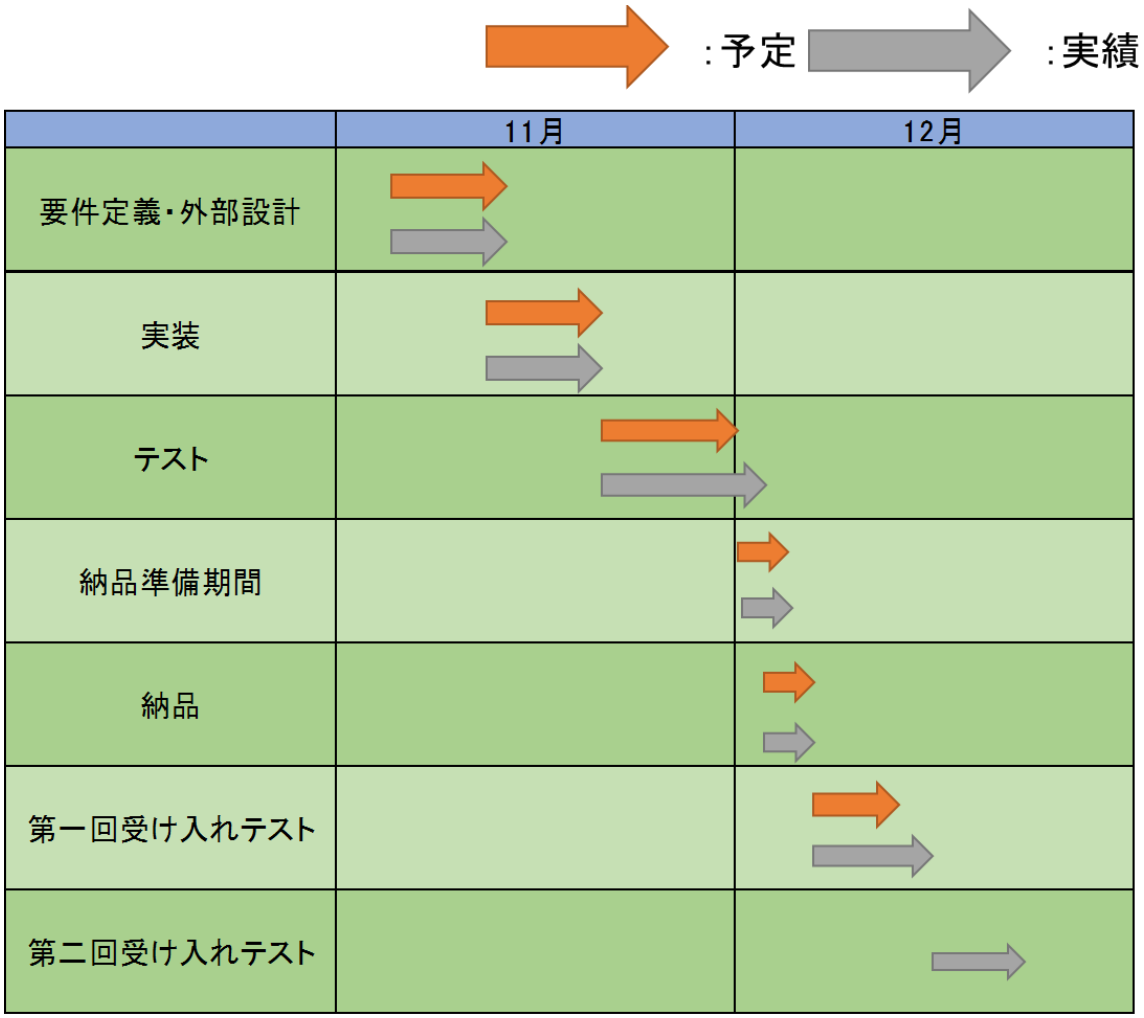
そのため、一部の機能の実装を見送った。実装を見送った機能とその概要を以下に示す。

- IRS のテーブル表示機能：IRS から取得できる情報について調査不足から一部の情報を正しく表示できなかった。顧客と相談し、第二次開発で改めて開発することとなった。
- データの条件抽出機能：Hadoop API の調査不足から条件付き抽出ができなかった。顧客と相談し第二次開発で改めて開発することとなった。
- テーブル作成機能：テーブル作成時のデータ型やデータ型の長さの指定などの詳細な要件定義が漏れており、受け入れテストで問題が発覚した。そのため、第二次開発で改めて要件定義からやり直すこととなった。

2.7.2 第二次開発：追加機能の開発

2.7 節で述べたように全体スケジュールが変更になったため、第二次開発のスケジュールも変更となった。第二次開発の変更後のスケジュールと実績を表 2.7 に示す。

表 2.7 第二次開発スケジュールと実績



第二次開発では第一回受け入れテストで問題が発生した。問題の内訳はマニュアルの不備が四点、データ移行中の進捗状況の表示機能の不具合一点であった。そのため、急遽修正を行い第二回受け入れテストを行い、12 月 17 日合格となった。第二回受け入れテストの実施はスケジュールで想定外だったため、追加の日数が必要となり第二次開発は一週間の延長となった。

また、第二次開発は第一次開発のレビューを受けて行われた。実際に開発を行った機能は第一次開発で実装を見送った機能や試験的に運用した際に得られたレビューへの対応など小規模なものであった。また、将来的な追加開発を容易にするためのプラグイン化対応も行った。

さらに、第一次開発におけるプロジェクト運営の失敗を踏まえ、コアタイムの実施や Redmine[8]、Git[9]などのツールのルール共通化によって管理体制の強化を行った。

第3章　市場調査及び技術調査

本章では著者が ETL ツール開発のために行った既存 ETL ツールの調査、及び Eclipse の調査について述べる。

3.1　ETL ツールの調査

2.3 節のシステム提案を行うために、既存の ETL ツールの調査を行った。本プロジェクトでは学生四人全員が ETL ツールに関する知識が全くなかった。そのため、システム提案を行うために ETL ツールそのものに関する調査から、開発する上で必要な既存 ETL ツールの特徴について調査を行った。

3.1.1　ETL ツールとは

ETL ツールとは DB や DWH からデータを抽出(Extract)し、必要な形式にデータを変換・加工(Transform)、移行先となる他の DB に格納(Loading)するツールであることがわかった。また、蓄積されたデータを活用する上で、ETL の工程が全体の 60~80%を占めることもあり[10]、ETL ツールによる工程の効率化が期待されている[3]。

3.1.2　既存 ETL ツールについて

IRS に対応した ETL ツールを開発するために既存 ETL ツールの特徴を調査した。一般的 RDB に対応した ETL ツールは Talend Open Studio[11]（以下、TOS）や JasperSoft ETL[12]など多数存在していることが調査の結果判明した。

1. Talend Open Studio について

TOS は Eclipse ベースで開発されているオープンソースの ETL ツールである。大きな特徴として GUI 上で移行元、移行先の DB やテーブルの対応関係を定義することが可能である。また、移行元、移行先の DB やテーブル、データの変換や加工などはユーザーが直感的に利用できるようにアイコン化されている。ユーザーはそのアイコンを指定ウィンドウにドラック＆ドロップし、対応関係を GUI 上で定義することで対象となる DB やテーブルに対して様々な機能が利用できるようになっている。

さらに、簡単な Java のコードを対応関係の中に埋め込むことも可能である。これにより、技術者であれば GUI でデータの流れを定義し、Java コードを埋め込むことでデータの抽出や変換が可能である。また、NoSQL 型の DB からもデータを抽出することができ、有償のコンポーネントを組み込むことで Hadoop 環境での差分更新なども行うことができる。

しかし、多機能ではあるが IRS に対応した ETL ツールではないため IRS を利用したビッグデータ分析環境では用いることができない。

2. JasperSoft ETL について

JasperSoft ETL は BI ツールとしても利用が可能なオープンソースの複合ツールでもある。TOS と同様に GUI 上で移行元、移行先の DB やテーブルの対応関係を定義することが可能である。

さらに、OracleDB や IBM の DB2 など様々な DB に対応しており、Hadoop 環境においても利用することが可能である。しかし、TOS と同様に IRS には対応していないため、IRS を利用したビッグデータ分析環境では用いることができない。

3.1.3 既存 ETL ツールの調査結果

3.1.2 節の調査から既存 ETL ツールの大きな特徴としてプログラミングではなく、GUI を用いてマウス操作でデータ移行が可能となっている点が挙げられる。具体的には移行元、移行先となる DB や加工・変換の関数が直感的に利用できるようにアイコン化されている。そして、マウスでアイコン同士を関連付けることによってデータの移行が可能である。これにより、ユーザーは SQL や DB の専門的な知識なしにデータの移行が可能となっている。

したがって、NET ツールにおいても GUI を用いたマウス操作でデータ移行を可能にする。そのために、移行元、移行先 DB やテーブル、加工、変換などの機能をアイコン化する。また、各機能を利用するために必要な入力は必要最低限とし、ユーザーへの負担を減らす。

これにより、ユーザーが専門的な知識を持っていなくてもデータ移行が可能になると考えられる。

3.2 Eclipse の調査

本プロジェクトではユーザーの習熟コストとプロジェクトにおける開発コストを削減するために Eclipse プラグインとして開発する。そこで、Eclipse の構造や GUI のライブラリについて調査し、Eclipse プラグインとして NET ツールが採用する開発方針を決める。

3.2.1 Eclipse について

Eclipse[13]とは 2001 年に IBM によってオープンソース化された Java ベースの統合開発環境（以下、IDE）である。大きな特徴として拡張が容易な点が挙げられる。そのため、拡張によっては別の用途として利用することができる。例として Eclipse ベースの ETL ツール TOS などが挙げられる。

3.2.2 Eclipse の構造について

調査の結果、Eclipse はランタイムカーネルを除くすべてがプラグインによって構成されていることがわかった[14]。そのため、独自に開発したプラグインであっても Eclipse の他のプラグインと同等に Eclipse に統合することが可能である。さらに、既存のビューやエディターに対しての拡張が可能であり、開発コストを削減することが可能である。

よって、NET ツールでは既存の Eclipse のビューやエディターを拡張して開発を行う。これにより、一から開発する必要がなくなり開発コストを削減することが可能である。

3.2.3 Eclipse の GUI について

Eclipse の GUI は主に Standard Widget Toolkit(以下、SWT)とその拡張版とも言える JFace によって構成されている[13]ことが調査の結果判明した。

SWT は Eclipse Foundation により管理されており、SWT を利用することで Eclipse デザインと親和性が高い GUI を作ることが可能となる。しかし、SWT は拡張性に乏しい面があった。

そこで、SWT で NET ツールを実装することは難しいと判断し、JFace についてさらなる調査を行った。調査の結果、JFace は SWT に Model-View-Controller(以下、MVC)[15]パターンを取り入れたライブラリであることが判明した。そのため、拡張が想定される NET ツールにおいても拡張性は十分であると判断し、JFace を積極的に用いて実装することとした。

第4章　NET ツールの GUI 設計

本章では 3 章の調査を基に筆者が主に行った NET ツールの GUI 設計について述べる。

4.1　NET ツール利用ストーリーの作成

顧客への画面イメージ図の提案を漏れ無く行うために、典型的な利用ストーリーを作成した。なぜならば、機能毎に画面イメージ図を作成した場合に機能間の画面イメージ図が存在せず、顧客と想定のズレが発生しやすいからである。よって、利用ストーリー作成することで、画面イメージ図の提案における顧客との想定のズレを軽減できると考えられる。利用ストーリーの主な流れは以下のとおりである。

1. 入力、出力先となる DB に接続する
2. 接続した DB からテーブルを取得する
3. テーブル定義を参照した上で、移行するテーブルを選択する
4. 選択したテーブルのアイコンを作業ウィンドウにドラック＆ドラップする
5. 抽出、加工アイコンを作業ウィンドウにドラック＆ドラップする
6. 入力、出力先のテーブルと関数を紐づける
7. 抽出アイコンをダブルクリックし、抽出条件設定画面を開く
8. 移行元データベースから抽出する条件を設定する
9. 加工関数をダブルクリックし、カラム間を紐付ける
10. 5~9 で対応関係が取れた場合はプロジェクトを実行する

顧客への提案の際には、さらに成功、失敗ケースなどを追加して、GUI ストーリーとして合意となった。

4.2　基本的機能と GUI の対応付け

4.1 節から NET ツールが提供する基本的な機能を Eclipse から GUI で操作可能にするために必要な機能を洗い出した。GUI に必要な機能を以下に示す。

1. プロジェクトを管理、表示する

任意の ETL タスクをその他の ETL タスクと区別するために、ETL タスクを Eclipse のプロジェクト単位で管理、ユーザーに表示する機能。

2. 利用可能な抽出、変換関数を表示する

利用可能なデータ抽出機能、変換機能を関数とし、ユーザーには関数をアイコン化して直感的に表示する機能。

3. データベースへの接続を行う

データベース間のデータ移行を行うためにデータベースへの接続を行う必要があり、GUI で接続をガイドする機能。

4. テーブル情報を表示する
ユーザーがデータベースに対する専門的な知識なしにデータ移行を行うために、NET ツール上からテーブル情報を確認する機能。

5. テーブルを新規に作成する
移行先データベースに適切なテーブルがなかった場合に、NET ツール上からテーブルを作成する機能。

6. テーブルと関数の対応関係を定義する
ユーザーへ直観的に対応関係を示すため、GUI をもってマウス操作でテーブルや関数間の対応関係を定義する機能。

7. データベースからテーブルを抽出する条件を定義する
ユーザーが SQL などの条件文を知らない場合においても、データの条件抽出を行えるように簡単な条件を設定できる機能。

8. テーブル間でカラムの対応関係を定義する(※)
※同プロジェクトメンバーの HU　BEIQI が担当するため、本報告書では論述しない。

4.3　　画面イメージ図の作成

4.1 節、4.2 節を踏まえ、NET ツールの画面イメージを作成した。本節では作成した画面イメージの中で特に 4.2 節の「6.テーブルと関数の対応関係を定義する」機能について述べる。その他の画面イメージについては付録の「納品報告書」に含める。

「6.テーブルと関数の対応関係を定義する」機能の画面イメージ図を 2.2 節顧客の要求と 3 章の既存 ETL ツール及び Eclipse の調査から作成した。作成した画面イメージを図 4.1、図 4.2 及び図 4.3 で示す。

イメージ図

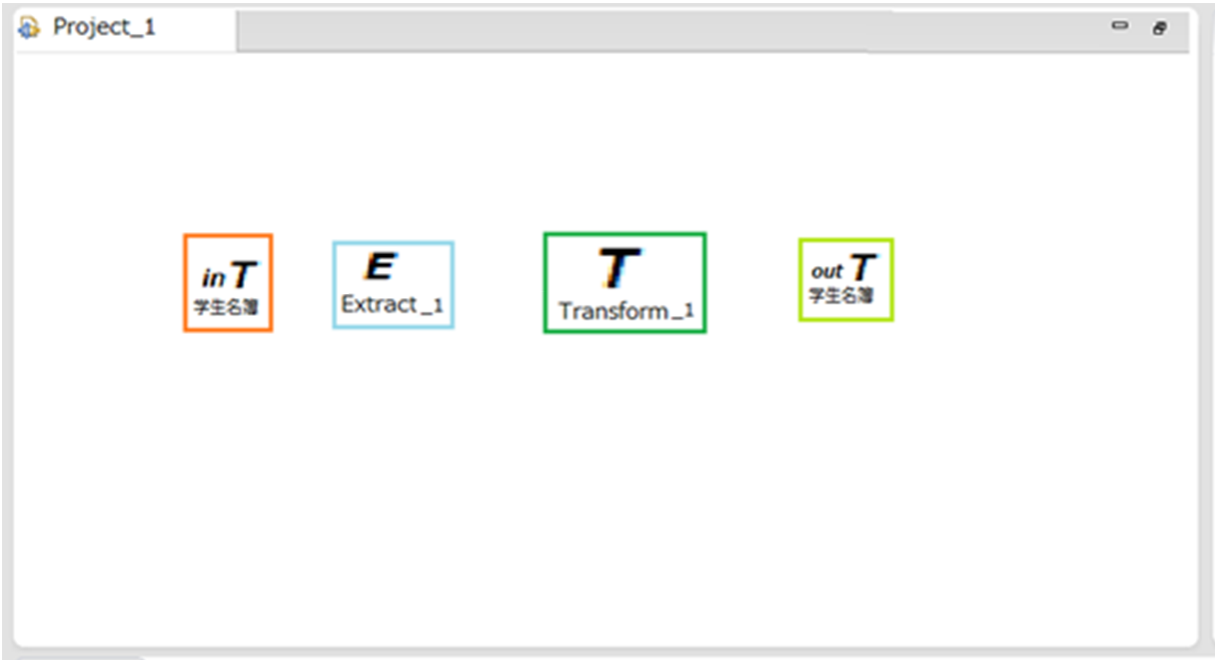


図　4.1 対応関係を定義する機能(1)

図 4.1 は左から移行元のテーブル、抽出関数、変換及びカラム間の関係定義関数、移行先のテーブルが GUI 上に表示されている画面イメージ図である。図の中央にある四色の四角はそれぞれ、橙色は移行元のテーブル、水色は抽出関数、緑色は変換及びカラム間の関係定義関数、黄緑色は移行先テーブルを表している。ユーザーはこれらを必要に応じて、「1.プロジェクトを管理、表示する」機能と「2.利用可能な抽出、変換関数を管理、表示する」機能から選択して GUI 上に表示する。なお、図 4.1 では移行元テーブルとして IRS の「学生名簿」テーブルが選択され、移行先テーブルとして PostgreSQL の「学生名簿」テーブルが選択されている。

ここで、「1.プロジェクトを管理、表示する」機能について述べる。この機能の画面イメージ図を図 4.2 に示す。この機能では一つの ETL タスクをプロジェクト単位で管理する。そのため、プロジェクトを最上位にして、その下に移行元 DB と移行先 DB が階層構造で表現されている。その中で、ユーザーは移行元、移行先のテーブルを DB の持つテーブルリストから選択することができる。「2.利用可能な抽出、変換関数を管理、表示する」機能についても同様に関数を選択することができる。

図 4.2 プロジェクト管理、表示する機能の画面イメージ図

そして、GUI 上に表示されたテーブルと関数を線で結ぶことで対応関係を定義できる。線で結んだ図を図 4.3 に示す。図 4.3 では移行元テーブルと抽出関数が線で結ばれており、移行元テーブルの「学生名簿」に対してデータの抽出を行うことが定義されている。同様に、抽出関数と変換及びカラム間の対応関係定義関数、移行先テーブルの「学生名簿」が線で結ばれている。これにより、抽出したデータを変換及びカラム間の対応関係定義関数で移行先テーブルの「学生名簿」に格納することが定義されている。これにより、図 4.3 では移行元テーブル「学生名簿」から移行先テーブル「学生名簿」へのデータの流れを GUI 上で確認できる。

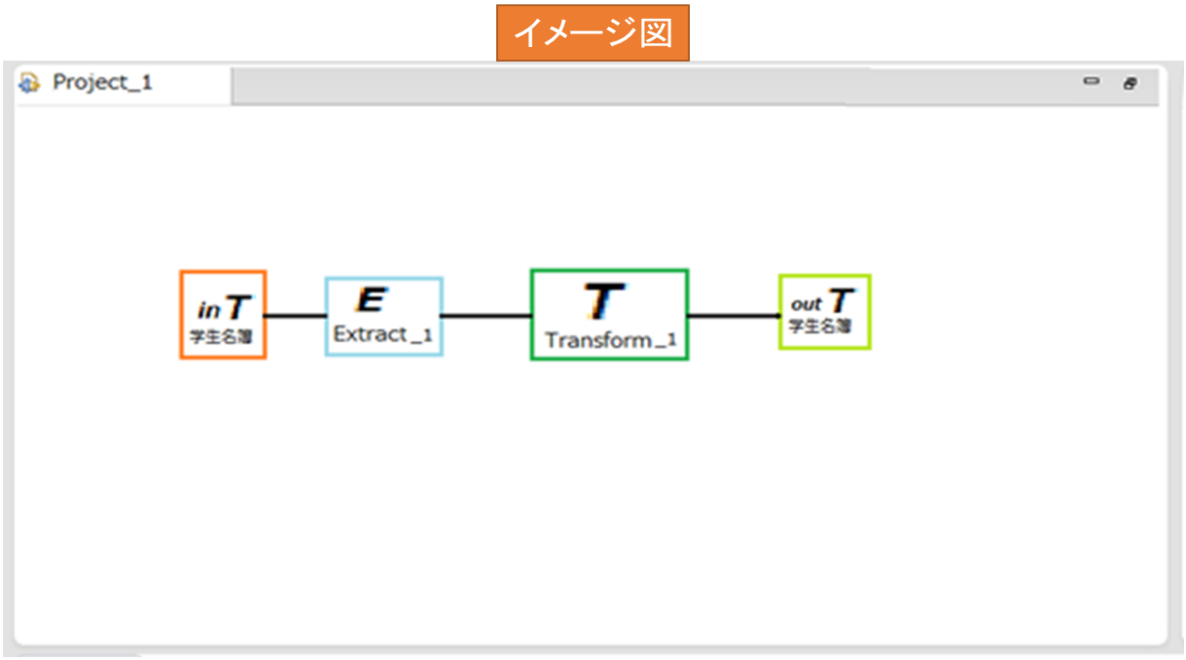


図 4.3 対応関係を定義する機能(2)

4.4 画面イメージ図の提案

4.3 節で作成したイメージ図を 4.1 節で作成した利用ストーリーに沿って顧客に提案した。顧客からは「さらに失敗ケースについてエラー文を詳細に表示して欲しい」や「テーブル情報の表示の際にはテーブル定義だけではなく、サンプルデータも表示して欲しい」などの要望があった。

そこで、顧客からの要望を受け入れ、「DB への接続を失敗した場合、エラー文を表示する」など特に外部との接続が発生する場合にエラーを表示することとなった。さらに、「テーブル情報の表示の際にはテーブル定義情報とサンプルデータを一行表示する」などとした。

4.5 GUI の内部設計

4.2 節で対応付けた基本的な機能を Eclipse で操作可能にするため、内部設計を行った。調査結果から、Eclipse のコンポーネントを拡張することで開発コストを削減する。さらに、Eclipse の GUI ライブラリである JFace を利用して実装することで拡張性を持たせつつ、Eclipse デザインに沿ったまま NET ツールの実装を行う。

1. プロジェクトを管理、表示する

まず、本機能をプロジェクト管理ウィンドウと呼ぶ。プロジェクト管理ウィンドウは筆者が設計した TreeView クラスで管理される。TreeView クラスのクラス図の一部を図 4.3 に示す。TreeView クラスは JFace の org.eclipse.jface.viewers.TreeViewer（以下、TreeViewer）と Has-a の関係である。また、TreeViewer から提供されている Controller である、ITreeContentProvider や LabelProvider を拡張して実装することで開発コストを削減している。また、GUI からの操作を Action としてセットしているため、新たな操作に対しても Action として容易に追加開発が可能である。

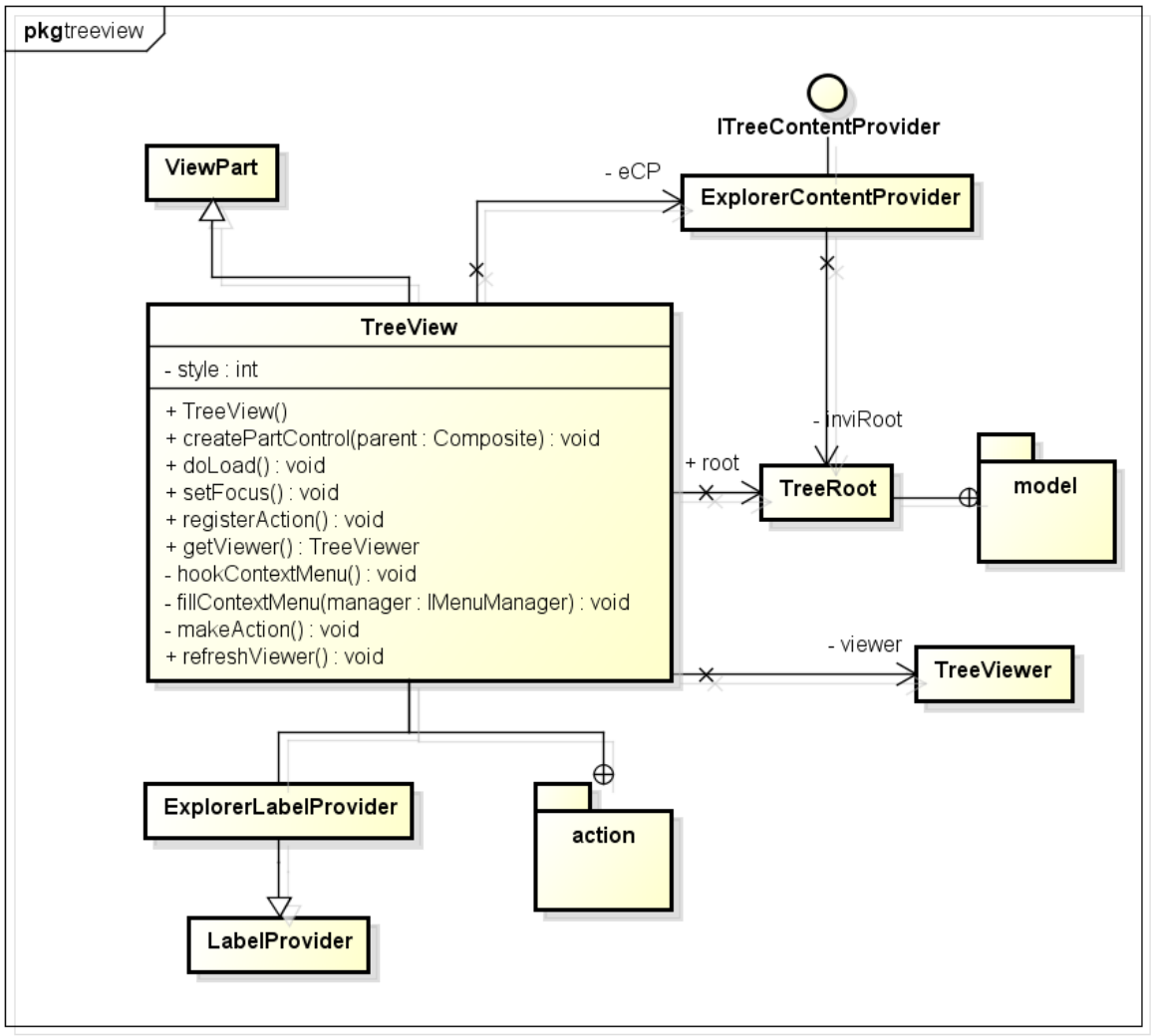


図 4.4 TreeView クラス図(一部)

さらに、Model クラスについても工夫を行った。Model クラスの一部を図 4.4 に示す。Model クラスは Composite パターンを利用して実装されるため、プロジェクト管理ウィンドウにおける全ての Model クラスは TreeNode か TreeLeaf を継承して実装されることになる。そのため、将来的な拡張開発が容易である。例えば、Model クラスの追加を行う場合には、他 Model クラスと同様に継承することで容易に実装、追加ができる。また、拡張開発を行う場合には親クラスを拡張することで全ての Model クラスに変更を適用することが可能である。

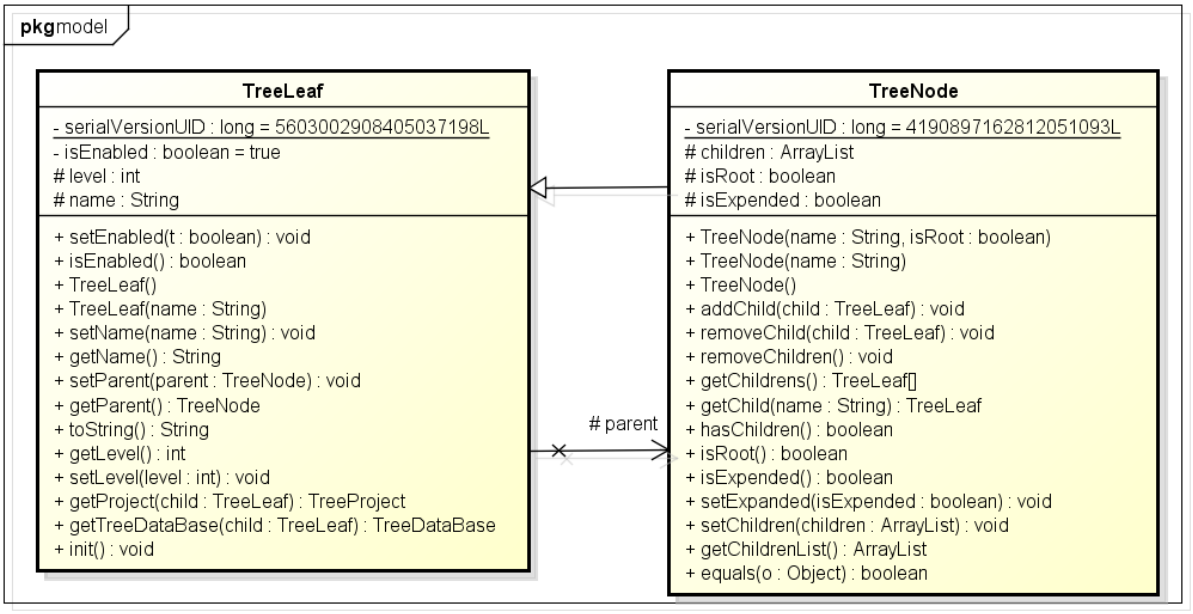


図 4.5 TreeView の Model クラス(一部)

2. 利用可能な抽出、変換関数を表示する

1.と同様の設計を行った。

3. データベースへの接続を行う

本機能をデータベース接続ウィンドウと呼ぶ。データベース接続ウィンドウは Eclipse の org.eclipse.ui.forms.editor(以下、Editor)を拡張し、Editor のサブクラスである FormEditor として実装を行う。

本プロジェクトにおいては IRS、PostgreSQL のそれぞれに対して適した接続ウィンドウを表示する必要がある。そのため、筆者は FormEditor の Is-a として親クラスを設計し、その子クラスとして IRS、PostgreSQL に対する接続ウィンドウを設計した。そのため図 4.5 のような継承関係になっている。また、FormEditor の必須の引数である IEditorInput に関しても同様に実装を行った。

これにより、IRS と PostgreSQL それぞれに対して 1 から接続ウィンドウを作る必要がなくなり、開発コストを削減することができた。また、新たにデータソースを追加する場合においても親クラスを継承することによって、容易に追加開発を行うことが可能になった。

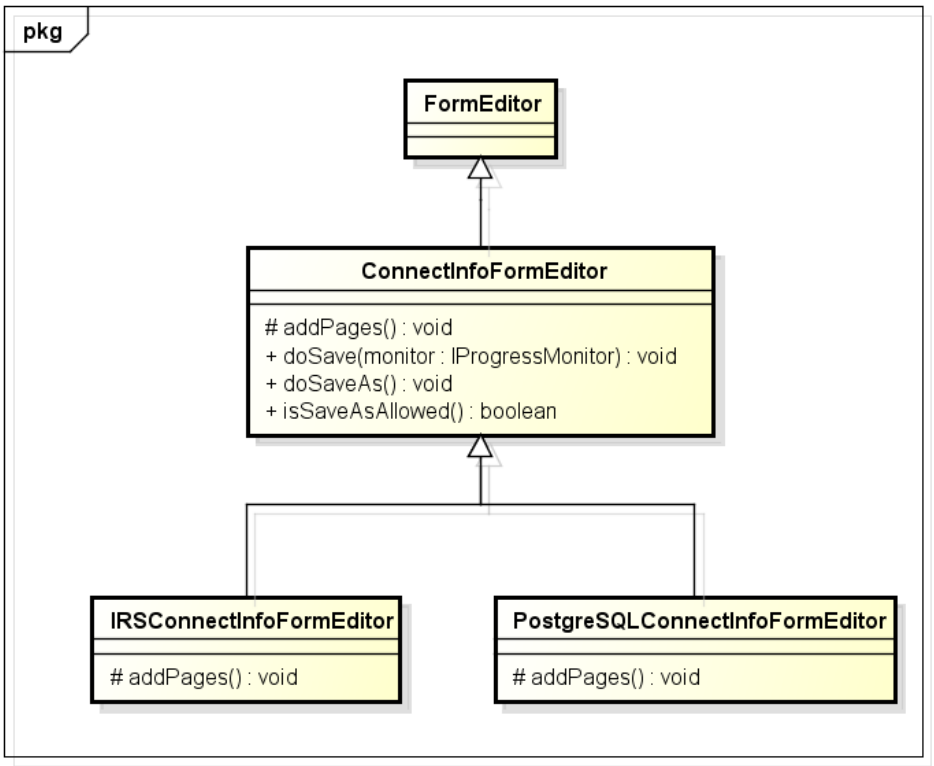


図 4.6 データベース接続ウィンドウのクラス図（一部）

4. テーブル情報を表示する

本機能をテーブル情報確認ダイアログと呼ぶ。テーブル情報確認ダイアログはプロジェクト管理ウィンドウの org.eclipse.jface.action.Action(以下、Action)とすることで、ユーザーからの操作を Eclipse から受け取ることができように設計した。これにより、OS の違いによる操作の違いを Eclipse で吸収しつつ、テーブル情報を表示するために必要な情報を Action の引数として取得できるようになった。

また、JFace の org.eclipse.jface.dialogs.Dialog(以下、Dialog)を継承して実装することとした。これにより、ボタンやテーブルなどの GUI を Dialog 上に SWT で作成するだけで実装できるようになった。

5. テーブルを新規に作成する

4.と同様の設計を行った。

6. テーブルや関数間の対応関係を定義する

本機能を作業ウィンドウと呼ぶ。作業ウィンドウは Editor を拡張し、org.eclipse.gef.ui.parts.GraphicalEditorWithPalette(以下、GE)を利用して実装する。GE は Eclipse が提供している Graphical Editing Framework(以下、GEF)の 1 つであり、MVC パターンにおける View の多くを提供している。よって、筆者は Model クラスと Controller クラスを作成し、一部の View を変更することで、GUI を実装することできるようになった。

特にモデルクラスについて詳細に論述する。作成した Model クラスのクラス図を図 4.6 に示す。図 4.6 の通り Model クラスは Composite パターンを利用して実装する。そのため、プロジェクト管理ウィンドウと同様に追加開発が容易である。したがって、Extract クラスや Input クラスは図 4.6 のように NodeElement を継承することで容易に追加ができる。

また、Model クラスの変更を View に通知するために Java.beans.PropertyChangeSupport を全モデルクラスが利用している。

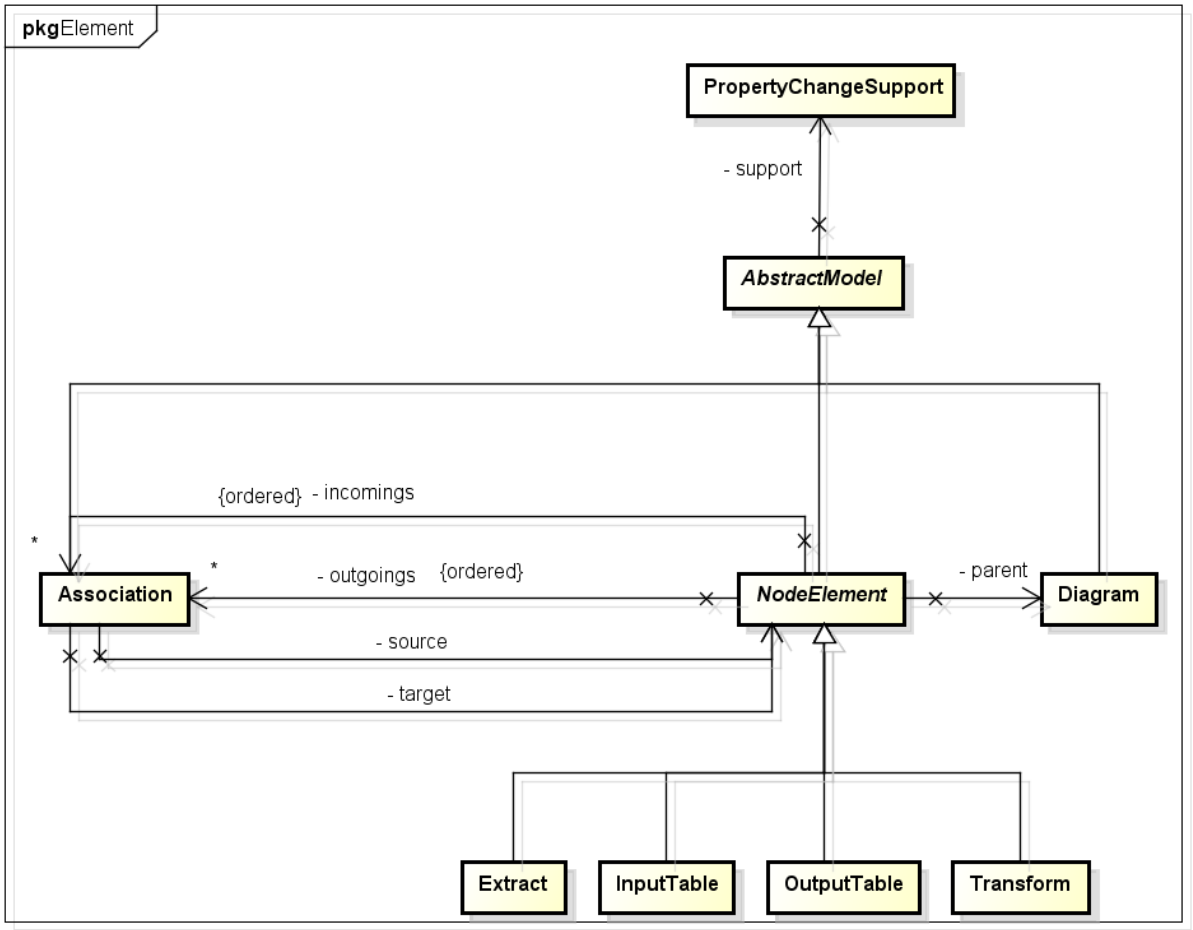


図 4.7 作業ウィンドウのモデルのクラス図(一部)

7. データベースからテーブルを抽出する条件を定義する

本機能を条件抽出ウィンドウと呼ぶ。条件抽出ウィンドウは Editor を拡張して実装されている。しかし、作業ウィンドウでの紐付けの状況により動的に条件抽出の対象となるテーブル情報を変更し、その変更を他クラスに通知する必要がある。そのため、SWT の Table クラスではなく、JFace の org.eclipse.jface.viewers.TableViewer（以下、TableViewer）を利用して実装することとした。理由としては、前述の通り JFace であれば MVC パターンを容易に取り入れることができ、動的に表示の変更を行うことができるからである。

条件抽出ウィンドウのクラス図の一部を図 4.7 に示す。図 4.7 の通り JFace の Viewer を

利用しているため、プロジェクト管理ウィンドウと類似したクラス図となる。相違点としてはテーブルの変更を管理する MyTableModifier クラスと条件文の入力をアシストする Dialog クラスがあるが挙げられる。また、作業ウィンドウの Model クラスである Extract クラスに Editor の引数である IEditorInput クラスを持たせることで、変更があった場合に PropertyChangeSupport の仕組みを用いて変更を通知している。

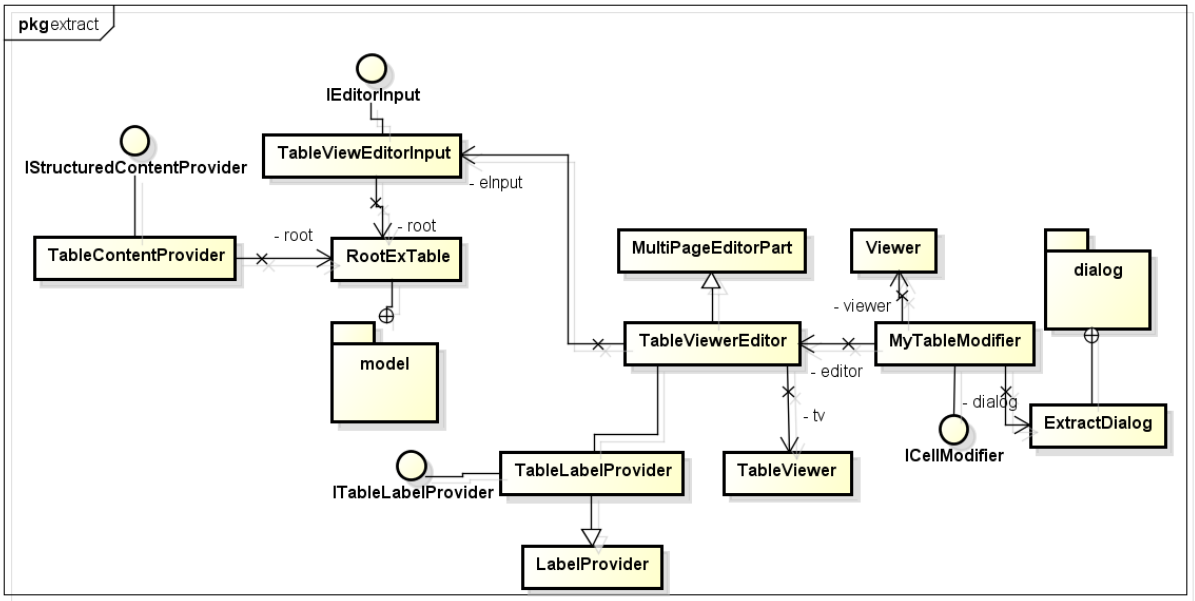


図 4.8 条件抽出ウィンドウのクラス図(一部)

これにより、MVC パターンで GUI を実装することができ、将来的な拡張性を確保した上で必要な機能の実装が可能となった。また、同プロジェクトメンバーとの結合を考慮して、中間モデルクラスを定めることで仕様変更に対応しやすい設計にした。

第5章　NET ツールの GUI 実装

本章では筆者が主に担当した NET ツールの GUI について、実装した実際の画面とイメージ図との差分について述べる。

5.1　プロジェクト管理ウィンドウ

プロジェクト管理ウィンドウは 4.2 節における「プロジェクトを管理、表示する」を行うための GUI である。同様に実装される「利用可能な抽出、変換関数を表示する」についても本節で述べる。

作成したイメージ図と第一次開発での実装の結果を図 5.1 に示す。イメージ図はプロジェクト、移行元と移行先 DB、IRS と PostgreSQL が階層関係で表示されるように作成をした。第一次開発では想定通りに階層関係が表現できた。実際に図 5.1 ではプロジェクトの Project_1、移行元 DB の inputDB、IRS と階層関係となっている。

図 5.1 イメージ図と第一次開発での画面図

さらに、第二次開発では IRS のテーブル名の表記を「テーブル名」から「DB 名.テーブル名」に変更して欲しいとの要望があった。理由としては、異なる DB に属する同テーブル名のテーブルを区別できないからである。また、PostgreSQL の場合はデータベースを指定して接続するので、同様の問題が発生しない。そのため、第二次開発では NEC と協議し、IRS のテーブル名の表記だけを変更した。図 5.2 に第一次開発と第二次開発の実装した画面を示す。第二次開発では第一次開発と比較して IRS のテーブルリストにあるテーブル名が「users」から「auction.users」に修正されている。

図 5.2 第一次開発、第二次開発での画面図

また、プロジェクト管理ウィンドウ内でプロジェクトや移行元 DB の名前の横に、それぞれ対応した画像を表示する予定となっていた。図 5.1 のイメージ図では黒▼で表現されている。しかし、第一次開発の遅延や優先度が低いことから開発を見送り、暫定的にフォルダの画像を表示することとなった。

5.2　　データベース接続ウィンドウ

データベース接続ウィンドウは 4.2 節における「データベースへの接続を行う」を行うための GUI である。本節ではデータベース接続ウィンドウの実装画面とイメージ図との比較を行う。

作成したイメージ図と IRS 実装の結果を図 5.3 に示す。イメージ図では ID、PASS、Database、Port の四種類の情報を入力する想定であった。しかし、IRS の Hadoop API を利用するためには、四種類の情報では不足していることが判明した。そのため、図 5.3 の IRS 接続ウィンドウでは入力する情報を IRS の仕様に基いて追加した。

また、イメージ図ではデータベースの持つテーブル情報の取得を行う「get table」ボタンがある。これは、データベースへの接続とデータベースの持つテーブル情報の取得を別々に行う想定でイメージ図を作成したためである。しかし、データベースへの接続と同時にテーブル取得を行う仕様に変更となったため、テーブル取得ボタンを IRS 接続ウィンドウでは削除した。

図 5.3 イメージ図と IRS 接続ウィンドウの比較

5.3　テーブル情報確認ダイアログ

テーブル情報確認ダイアログは 4.2 節における「テーブル情報を表示する」を行うための GUI である。本節ではテーブル情報確認ダイアログ実装時の工夫、イメージ図との比較を行う。

作成したイメージ図を図 5.4 に、実装の結果を図 5.5 に示す。イメージ図ではプライマリキー（以下、PK）、カラム名、データ型、長さ、NOT NULL 制約の五種類のテーブル定義情報とテーブルの実データを Table Example として一行表示することとなっていた。実装した画面でも同様の情報を表示することができた。

しかし、イメージ図では NOT NULL 制約をチェックボックスで表示することとしていたが、実際には図 5.5 の通り文字で表示するようになった。これは第一次開発での遅延が発生したため、工数を削減できる文字の表示となった。

さらに、IRS に対する調査不足による要件定義漏れや例外処理の抜けなどがあり、第一次開発での機能完成を見送った。そのため、第二次開発において、再度要件定義を行い「IRS のサンプルデータは表示しない」や「サンプルデータがない場合は空欄を表示する」などの要件を明確にし、実装を行った。

イメージ図

図 5.4 テーブル情報確認ダイアログのイメージ図

実装したテーブル情報確認ダイアログ

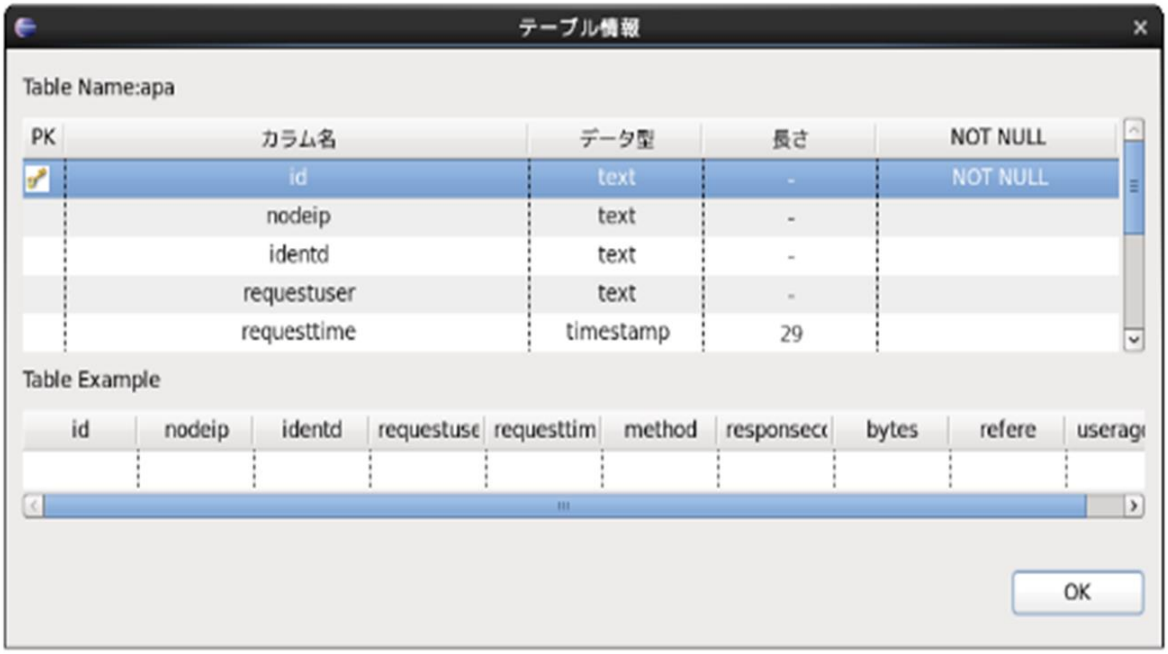


図 5.5 テーブル情報確認ダイアログの実際の画面

5.4 テーブル作成ダイアログ

テーブル作成ダイアログは 4.2 節における「テーブルを新規に作成」を行うための GUI である。実装については 5.3 節とほぼ同様であるため、本節ではイメージ図との比較、実装後の修正などに関して述べる。

図 5.6 に作成したイメージ図を、図 5.7 に実装の結果を示す。イメージ図ではテーブル名を指定し、作成するカラムに対して PK 制約、カラム名、データ型、長さ、NOT NULL 制約を指定できることとなっていた。また作成したカラムは削除ボタンで随時消すことができるとした。

しかし、PostgreSQL の仕様の調査が不足しており、実装時に図 5.7 の通り UNIQUE 制約を追加することとなった。また、機能としては一部の型に対して長さが指定できないバグが存在し、第二次開発において指定ができるように修正を行った。

イメージ図

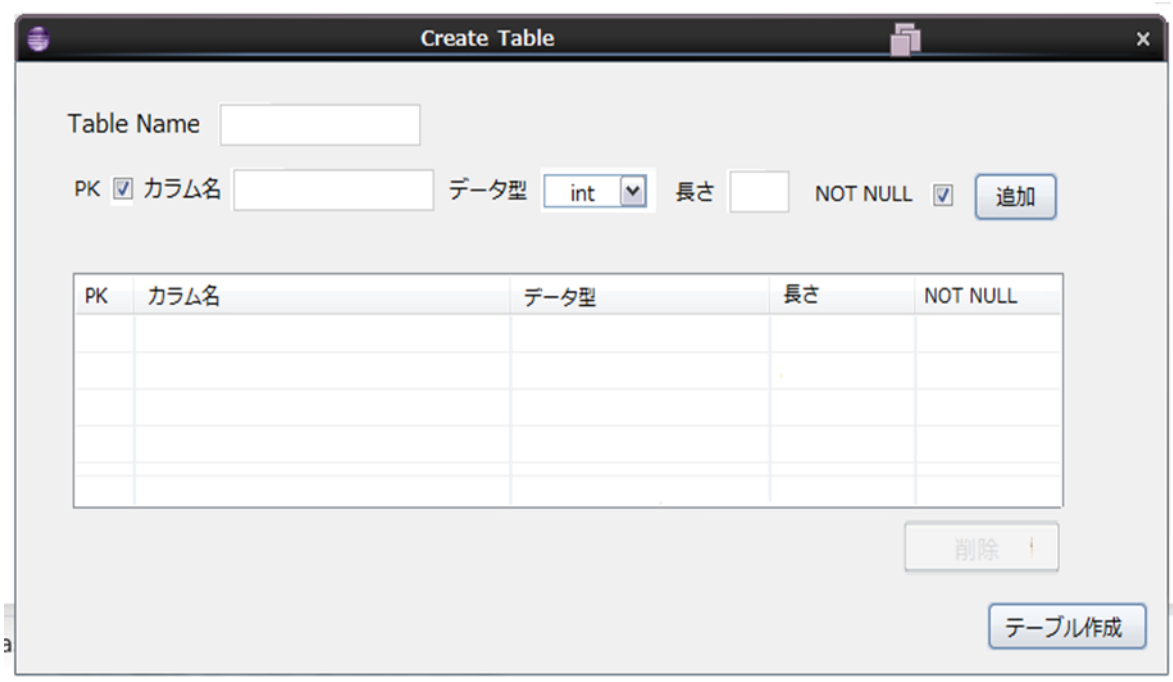


図 5.6 テーブル作成ダイアログのイメージ図

実装したテーブル作成ダイアログ

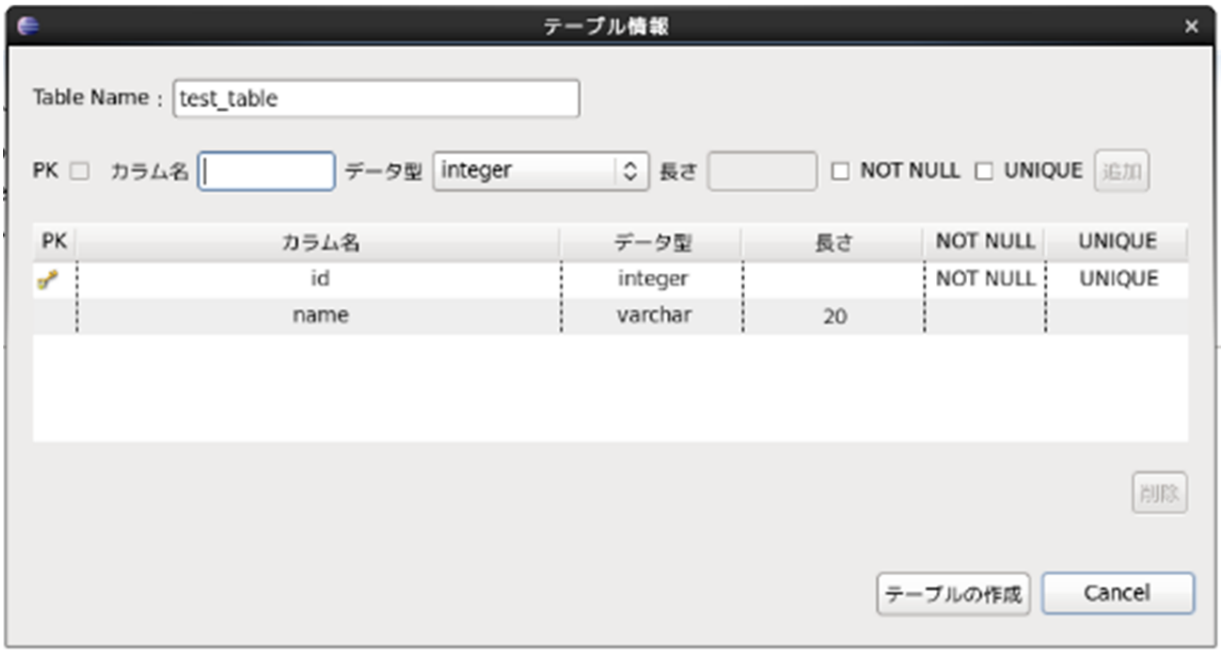


図 5.7 テーブル作成ダイアログの実装した画面図

5.5　　作業ウィンドウ

作業ウィンドウは 4.2 節における「テーブルと関数の対応関係を定義する」を行うための GUI である。本節では作業ウィンドウ実装時の工夫やイメージ図との比較を行う。

作成したイメージ図を図 5.8 に示す。4.3 節で前述の通りイメージ図では移行元 DB のテーブル、抽出関数、変換及びカラム間の関係定義関数、移行先 DB のテーブルの色を分けて表示する。さらに、5.1 節のプロジェクト管理ウィンドウからドラッグ＆ドロップすることで、作業ウィンドウ内で操作可能なアイコンとしてテーブルと関数を作業ウィンドウ内で操作可能にすることとした。

イメージ図

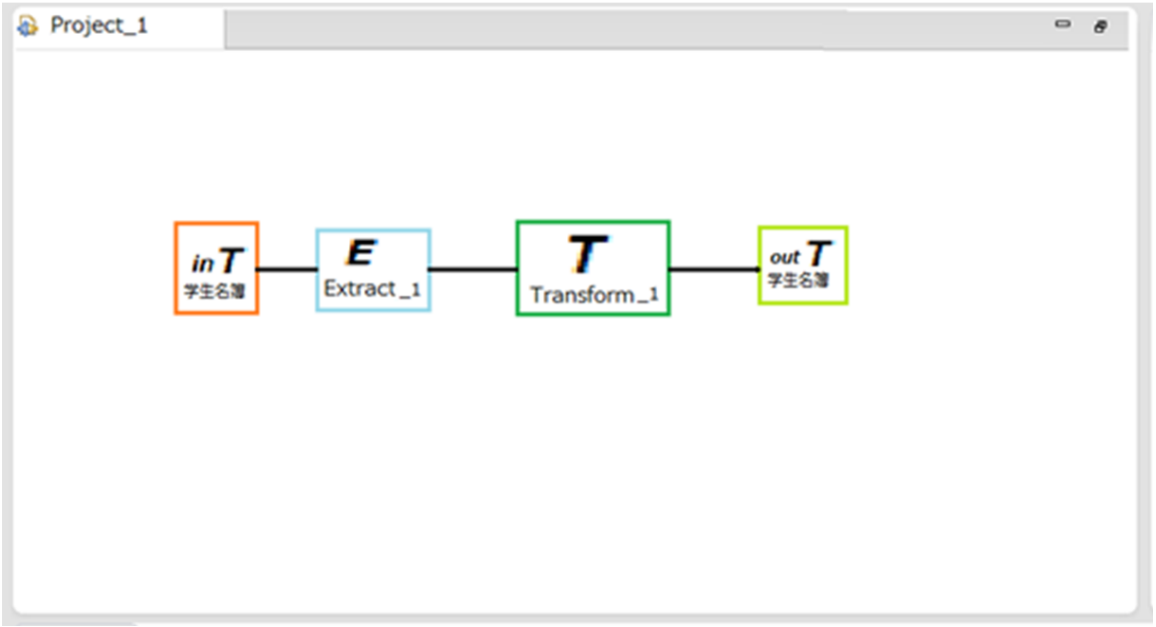


図 5.8 作業ウィンドウのイメージ図

そして、実装の結果を図 5.9 に示す。実装した画面ではイメージ図とテーブルと関数を色分けして表示するよう実装を行った。また、プロジェクト管理ウィンドウから作業ウィンドウ内にドラッグ＆ドロップできるようにもした。さらには、Eclipse の標準ショートカットである Delete キーによるアイコンや対応付けのための線の削除、Ctr+Z による Undo も行うことができるようにした。

しかし、第一次開発の遅延発生に伴い、テーブルと関数を表現するアイコンに関してイメージ図とは違った実装になった。色に関しても移行元 DB のテーブルは赤色、抽出関数は青色、変換及びカラム間の対応関係定義関数は緑色、移行先 DB のテーブルは灰色となった。ただし、アイコンの描画クラスを Model クラスと分けて作成しているため、将来的な変更は可能である。

作成した作業ウィンドウ

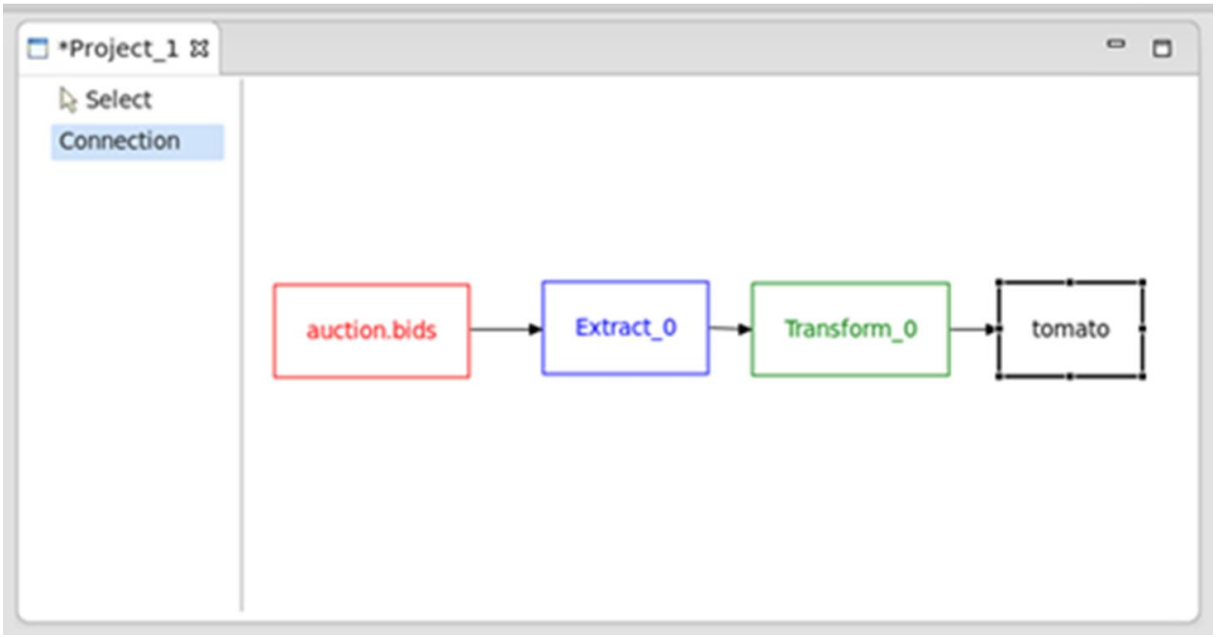


図 5.9 作業ウィンドウの実装した画面図

5.6 条件抽出ウィンドウ

条件抽出ウィンドウは「データベースからテーブルを抽出する条件を定義する」を行うための GUI である。本節では条件抽出ウィンドウの実装画面とイメージ図の比較を行う。

作成した条件抽出ウィンドウのイメージ図を図 5.10 に、さらに抽出条件を設定するダイアログのイメージ図を図 5.11 に示す。

条件抽出ウィンドウイメージ図

図 5.10 条件抽出ウィンドウのイメージ図

図 5.11 抽出条件設定ダイアログのイメージ図

まず、作成したイメージ図では抽出条件ウィンドウがユーザーに表示される。ユーザーはデータ移行元のテーブルから抽出するカラムをチェックボックスで選択することができる。これにより、ユーザーは移行元 DB のテーブルの定義情報を確認しながら、必要なカラムだけを抽出することができる。

さらに、抽出条件のカラムにあるボタンをクリックすることで、図 5.11 の抽出条件設定ダイアログが表示され、各カラムに対して抽出条件を設定することができる。抽出条件として条件と値を設定することができる。図 5.11 では「次の値の間」の条件と「201400001」から「20150000」の値を設定している。この条件で抽出した場合は 20140001 から 20150000 までの値を持つテーブルのみが抽出される。

また、図 5.10 の下に「詳細記述」を設けた。詳細記述に記述された命令文は IRS に抽出条件として送られる。この時の命令文は IRS の仕様に基づくものである。これは専門の技術者が IRS に対して高度に複雑な条件を指定可能にするために設けた。

そして、条件抽出ウィンドウの実装結果を図 5.12 に、抽出条件設定ダイアログの実装結果を図 5.13 に示す。

図 5.12 条件抽出ウィンドウの実装した画面図

図 5.13 抽出条件設定ダイアログの実装した画面図

図5.12ではイメージ図と同様に抽出対象のチェックボックスと移行元DBのテーブル情報であるカラム名、データ型を表示するように実装を行えた。イメージ図の「詳細記述」はわかりやすいように「詳細条件の設定」と名称を変更し、実装を行った。

しかし、抽出条件設定のダイアログを表示する操作がボタン操作ではなく対象のカラムをクリックする操作へと変更になった。これは、ボタンの実装に想定以上の工数を費やす必要があり、カラムをクリックする仕様であれば工数を削減できるからである。

さらに抽出条件の設定ダイアログの抽出条件の仕様が設計時に曖昧であった。そのため、実装時には図5.13ように一つの値を入力し、入力値に対して「等しい」、「より大きい」、「より小さい」、「以上」、「以下」及びDBの抽出条件に使われる「IN」条件が設定できるとした。

第6章　　テスト

本章では筆者が行った単体、機能、総合テストについて述べる。また、テスト報告書は付録に含める。

6.1　　単体テスト

本プロジェクトでは例外処理の漏れや仕様との違いなどを発見し、ソフトウェアの品質を保つために単体テストを行う。本プロジェクトでは開発規模を考慮して、View クラスを除いたすべてのクラスの Public メソッドに対して Junit を用いて単体テストケースを作成する。

筆者は主に開発を担当した GUI の Model クラス、Controller クラスに対しての単体テストケースを作成し、単体テストを行った。作成したテストケースの一例を表 6.1 に示す。

表 6.1 単体テストケース一例

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
jp.net.treeview.model.TreeLeaf	TreeLeaf()	メソッド実行時に適切に初期化されていることを検査する。	適切に初期化されている。	適切に初期化されている。	○	2014/10/10
	TreeLeaf(String name)	初期化され、さらに引数がメソッド実行時にインスタンスにセットされているかを検査する。	適切に初期化され、セットされた変数は引数と等しい。	適切に初期化され、セットされた変数は引数と等しい。	○	2014/10/10
	setEnabled(boolean t)	引数がメソッド実行時にインスタンスにセットされているかを検査する。	セットされた変数は引数と等しい。	セットされた変数は引数と等しい。	○	2014/10/10
	isEnabled()	メソッド実行時のインスタンスにセットされている変数が、適切に取得できるかを検査する。	取得できた変数とセットされていた変数は等しい。	取得できた変数とセットされていた変数は等しい。	○	2014/10/10

筆者が作成したテストケースは第一次開発で 207 件、第二次開発においては 5 件を追加した 212 件であった。第一次開発においては 207 件のテストケースの全てを作成する必要があり、大きな時間を要した。しかしながら、第二次開発においては新規でテストケースを作成する必要があったのは 5 件であり、単体テストを容易に行うことができた。

また、単体テストケースの作成によって第二次開発における仕様変更での影響調査を容易に行うことができた。実際に仕様変更によるバグの作り込みを回避することができた。

6.2　　機能テスト

本プロジェクトでは実際に NET ツールを操作し、機能が顧客と合意された内容と相違なく動作するかを機能テストでテストする。

著者は第一次開発において機能テストの一部を行い、担当部分の機能テスト報告書を作成した。第一次開発において機能テストは 23 件あり、著者が担当したテストは 4 件であった。以下に機能テストケースの一部を示す。

機能テストケース例

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

表 6.2 機能テストケースの一例

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

機能テストにより、NET ツールが正しく動作するかどうかを検査することができた。これにより、実際にエラー処理などの異常系に対して正しくエラーが表示されないバグを機能テストで取り出すことができた。

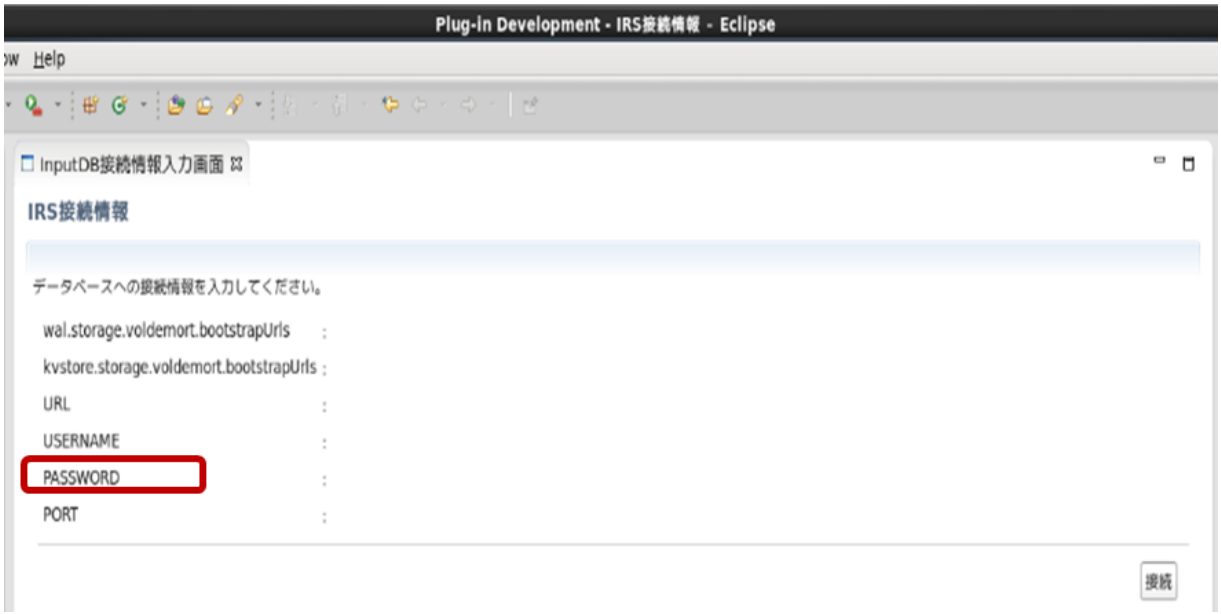
6.3　　総合テスト

本プロジェクトでは 4 章の GUI 設計に基づき、機能が実現されているかを総合テストにおいて検査する。著者は第二次開発において総合テストを担当し、4.2 節の利用シナリオに沿って NET ツールを動作させてテストを行った。総合テスト後は総合テスト報告書を作成し、顧客への納品物に含めた。総合テストのテストケースの一例を示す。

総合テストケース例

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

テスト内容：画面イメージ図と相違ないか確認する



※　1 件のバグ報告

バグ:2014/12/3 12:00 バグ PASSWORD が「PASSWORED」となっている。

対処:PASSWORD とする。

これにより、第二次開発では上記のような細かなバグが発見されたが、影響範囲は小さく無事終了することができた。

第7章　NET ツール GUI の評価

本プロジェクトではユーザーの負担を軽減することを主たる目的として NET ツールの開発を行った。そのため、顧客である NEC に評価アンケートを依頼し、NET ツールの有効性について検証を行う。

本章では特に筆者が担当した NET ツールの GUI について述べる。筆者は GUI に関するアンケートを作成し、NEC に回答を依頼した。これにより、NET ツールの GUI がユーザーの負担を軽減する上で障害になっていないかを検証し、将来的な開発の参考とする。

7.1　アンケート項目

アンケート項目を表 7.1 に示す。問 1~7 は選択回答とし、選択回答は基本的には四段階評価であるが、問 1 と 7 に関しては 0 から 10%刻みの十一段階評価である。また、問 8 は自由記述である。さらに、全ての問に対して任意の自由記述欄を設け、顧客からの意見をより正確に反映できるようにした。

表 7.1 GUI のアンケート項目

	質問事項
1	(選択)実際出来上がったGUIと設計はどの程度一致していましたか
2	(選択)NETツールのデザインとEclipseデザインで違和感を感じることはありましたか
3	(選択)NETツールの操作はEclipseの操作と大きな違いがありましたか
4	(選択)NETツールの操作で違和感を覚えることはありませんでしたか
5	(選択)4で違和感を覚えると答えた場合、違和感を覚えた操作はどれですか
6	(選択)NETツールの操作は簡単でしたか
7	(選択)NETツールのGUIの満足度はどの程度ですか
8	(自由記述)GUIで改善してほしい点がありますか

なお、GUI に関するアンケートを含む、全体のアンケートについては本報告書の最後に付録する。

7.2　アンケートの結果及び考察

アンケートの結果及び結果から得られた考察について本節では特に、ユーザーの習熟コストや Eclipse との親和性について論述する。

本プロジェクトではユーザーの習熟コストを削減することを主な目的として開発を行った。そのため、NET ツールの GUI の操作が複雑で、他 ETL ツールと大きく異なってはいけない。この点において、アンケート項目 4,5,6 で検証を行う。

アンケートの結果は NEC から「高い評価」を得ることができた。「Eclipse のショートカットと NET ツールの操作との対応付けを行った」や「既存 ETL ツールと同様にマウスで対応関係を定義する」などの Eclipse や既存 ETL ツールの操作特性 NET ツールに取り入れたこと結果として、高い評価が得られたと推測される。

次に、NET ツールは開発コストを削減するために Eclipse プラグインとして開発を行った。そのため、Eclipse の上で自然に動作する必要があり、デザインとの親和性をアンケート項目 2,3,4 から検証する。

アンケートの結果は「それなりに良い評価」を得ることができた。本プロジェクトの開発期間や開発範囲を考慮すれば、十分な出来であるとコメントもいただいた。

しかしながら、「既存の製品と比較して、一部の描画が見劣りする」や「一部の操作が右クリックからしか実行できない」などのご指摘もいただいた。これに関しては、ご指摘の通りである。前者に関しては描画クラスを変更して既存製品とも見劣りしないレベルにする必要がある。後者に関しては操作に対応したボタンやメニューを追加する必要があると考えられる。

第8章　　振り返り

本章では第一次開発、第二次開発終了時に行ったプロジェクトに関する振り返りの中で、特に筆者が学んだことを述べる。

第一次開発ではチームメンバー間の理解度も浅く、各メンバーのプロジェクト像が曖昧なままプロジェクト全体がリーダーである斎藤に依存する形となった。この体制では個人レベルでは遅延や不具合を発見できても､プロジェクト全体としては発見が遅れてしまっていた。

そのため、リーダーが一度体調不良になった時にプロジェクトが一週間程度無管理状態になり、今回の大きな遅延での発見につながった。この経験から、プロジェクトメンバーであっても主体性を持ってプロジェクトへ参加し、プロジェクト全体として管理する構造が必要であると学ぶことができた。

第二次開発では、反省を踏まえて筆者は主体的にプロジェクトに参加した。また、他メンバーもプロジェクトに参加するように積極的に声掛けを行った。その結果、リーダーだけがプロジェクト全体について考えるという風潮はなくなり、プロジェクト全体に積極性が生まれたと感じられた。

一方で、第二次開発では仕事量が少ないのにも関わらずコアタイムが第二次開発期間に固定で設けられていた。これによって、他メンバーのためだけにコアタイムに参加する場合が発生してしまい、筆者のみならず全体のモチベーションの低下につながった。前回の反省点を改善するためとは言え、動的にコアタイムの量を変化させることを進言することでモチベーションの低下を防げたと考えられる。よって、二週間から一ヶ月の周期でプロジェクトを改善するために意見を出しあう場を作ることが必要であることを学ぶことができた。

これより、主体性を持ってプロジェクトに参加し、チームメンバー同士で話し合いプロジェクト体制を修正、改善する必要があることを強く感じた。

第9章　おわりに

本プロジェクトは 2014 年度研究開発プロジェクトとして NEC を顧客に「スケールアウト型データベース製品に関わる設計開発」というテーマでプロジェクトを行った。本プロジェクトはユーザーの負担を軽減することを主たる目的とし、IRS に対応した ETL ツールの開発を行った。筆者は主に GUI の開発を担当した。

NET ツールの GUI 開発にあたり、ユーザーの操作習熟コストやツールのデザインを考慮して、既存 ETL ツールや Eclipse の調査を行った。調査結果を踏まえて、Eclipse プラグインとして ETL ツールの GUI の設計、実装を行った。また、NET ツールの今後の拡張を考えて MVC パターンを取り入れ、拡張性の向上に努めた。

NET ツールの GUI 評価では NEC にアンケートを依頼し、想定との違いや設計時との違い、満足度などを回答していただいた。製品と比較して、一部見劣りする点もあったが、全体としては概ね高い評価を得ることができた。

評価を下げることになった点に関して、「矢印の描画が見劣りする」や「一部の操作が右クリックからしか実行できない」などのご指摘をいただいた。前者であれば、将来的に描画クラスを変更することで既存の製品とも見劣りしないツールが開発できると考えられる。後者については、操作に対応したボタンやメニューを作成することで対応できると考えられる。

謝辞

プロジェクトの遂行にあたり、日本電気株式会社システムソフトウェア事業部の皆様にはテーマの提供並びにご指導、ご協力を頂けましたことを深く感謝いたします。

指導教員の田中二郎先生にはプロジェクトの遂行並びに特定課題研究報告書の執筆において多くのご指摘を頂きました。さらには、研究開発プロジェクトの中間報告においてもご指導を頂き、誠にありがとうございました。

課題担当教員の天笠俊之先生には、システムの提案における専門的な知識や技術から中間報告発表のプレゼンに至るまでご指導を頂きました。プロジェクトの遂行においても、親身になってご指導いただきまして、誠にありがとうございました。

さらに、水谷先生には私が大学生の頃からご指導いただきました。特定課題研究報告書においても、ご協力いただきまして、誠にありがとうございました。

最後に、本プロジェクトメンバーである斎藤、HU、唐とは多くの議論を重ねることで、非常に有意義なプロジェクト活動をすることができました。特にプロジェクト設立当時にはメンバーに多大なご協力をいただきました。皆様、大変ありがとうございました。

参考文献

[1] 木原 洋一. *エンタープライズ向けクラウドの現状と展望 -効率性・俊敏性の追求からのビッグデータ利活用による価値創造-*, IEICE technical report Volume 111, Issue 408, pp.99-102, 2012.

[2] M. Ferguson. *Architecting A Big Data Platform for Analytics*, Intelligent Business Strategies, 36p, 2012

[3] A. Titirisca. *ETL as a Necessity for Business Architectures*, Database Systems Journal, Volume 4, pp. 3-12, 2013.

[4] M. L. Songini. *ETL*, Computerworld, Volume38, Issue 5, p.23, 2004.

[5] J. Anitha and M. S. P. Babu. *ETL Work Flow for Extract Transform Loading*, International Journal of Computer Science and Mobile Computing, Volume 3, Issue 6, pp.610-617, 2014.

[6] 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-*,The journal of Information Science and Technology Association, Volume 62, Issue 11, pp.484-489, 2012.

[7] 日本電気株式会社,InfoFrame Relational Store V3.1.

[8] Redmine.org, http://www.redmine.org/, [2014-12-26 accessed].

[9] Git, http://git-scm.com/

[10] V. Gour, S. S. Sarangdevot, G. S. Tanwar, and A. Sharma. *Improve Performance of Extract, Transform and Load (ETL) in Data Warehouse*, International journal on computer science and engineering, Volume 2, pp.786-789, 2010.

[11] Talend Open Studio, http://jp.talend.com/products/talend-open-studio [2014-10-28 accessed].

[12] JasperSoft ETL, http://community.jaspersoft.com/project/jaspersoft-etl,[2014-6-28 accessed].

[13] Eclipse, http://www.eclipse.org,[2014-10-28 accessed].

[14] J. D. Rivieres and J. Wiegand. *Eclipse: A platform for integrating development tools*, IBM Systems Journal, Volume 43, Issue 2, pp.371-383, 2004.

[15] G. Krasner and S. Pope. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80*, Journal of Object-Oriented Programming, Volume 1, Issue 3, pp.26-49, 1988.

付録

- 納品報告書
- 顧客アンケート

納品報告書

本報告書では、筑波大学研究開発プロジェクト「スケールアウト型データベース製品に関わる設計開発」の納品物について記載するものである。

第１イテレーションについての記述は節名または項名に「第１イテレーション」と記述した。

第２イテレーションについての記述は説名または項名に「第２イテレーション」と記述した。

節名または項名に「第１イテレーション」または、「第２イテレーション」の記述がないものは第１イテレーションと第２イテレーション共通の項目である。

第１章　　目次

第２章　開発構想フェーズ

本章では開発構想フェーズで作成した成果物について述べる。成果物は開発構想書である。

第１節　　開発構想書

本節、プロジェクト開始時に作成し合意した開発構想の内容を記述する。開発構想書からの引用である。

第１項　　開発概要

開発の背景

1.　　InfoFrame Relational Store

ビッグデータ活用が行われるようになった今日、大量のデータに対応するためにデータベースに高い性能が求められるようになってきた。従来から広く使われている RDBMS では、増え続けるデータに対応するためにスケールアウトすることは容易でない現状がある。

一方、”Key-Value 型データベース”など NoSQL と呼ばれる RDBMS 以外のデータベースは拡張性が高く、容易にスケールアウトすることが可能で大量のデータに対応できるという特徴がある。しかし、NoSQL は新しい技術のため、独自 API や専用のシステム設計技術を習得した専任要員が必要であるなど、導入の障壁は高い。また、既存のデータやソフトウェア・人材といった資産を NoSQL に対応させることは難しい。

日本電気株式会社様は”Key-Value 型データベース”の特徴である拡張性の高さと RDBMS の特徴である扱いやすさを兼ね備えたデータベースシステム”InfoFrame Relational Store”（以下、IRS）を開発された。

IRS は次のような特徴を持っている。

1. データ量の増加に対応できるスケールアウト性能
2. 国際標準規格である SQL が利用可能
3. Hadoop API を使用し大量のデータに高速でアクセス可能

2.　　IRS を用いたデータ分析環境概要

IRS はスケールアウト性能やトランザクションの信頼性を重視した機構のため RDB と比べ以下の様な特徴がある。

1. トランザクションの信頼性やスケールアウト性能を高めるために役割の異なる三種類のサーバが三層構造を構成している。そのため、システムに処理要求を送ってから、結果の出力が終了するまでの時間(以下、TAT)が一般的な RDBMS よりも長い
2. Between などの範囲選択の命令や集計処理など広範囲のデータを対象とした

処理では高性能を期待できない

3. IRS の出力は RDB と異なりビューではなく、別に参照専用の別テーブルと作成する必要がある

そのため、すべての RDBMS を IRS に置き換え使用することはできない。IRS の持つ”スケールアウト性能”や”Hadoop 連携”の利点を活かすためには、IRS に大量のデータを蓄積し、必要に応じてデータを他 DB やデータウェアハウス(以下、DWH)に移行し、移行先で分析するというシステムが適している。

3. ETL (Extract/Transform/Load)

ETL の E は Extract (抽出)、T は Transform (変換)、L は Loading (格納)のことで、分析ツールが必要とするデータを、様々なデータソースから抽出し、適切な形式に変換し、DWH に格納する役割を持つツールのことである。

ETL ツールの機能として主なものを以下に述べる。

- 抽出機能(Extract)： 統合すべきデータを管理しているさまざまなデータベースやファイルにアクセスし、データを抽出する機能
- 変換機能(Transform)： 複数のデータソースから抽出して集めたデータを統合し、必要なコード変換や変更及び加工処理をデータ定義に沿って行う機能
- クレンジング機能： 基幹系データのもつ参照整合性エラーなどを修正して、データウエアハウスへのロードを可能にする機能
- ロード機能(Loading)： 変換、加工を終えたデータを DWH やデータベースに流し込む機能

ETL ツールを使用するとデータ移行処理が効率化され、コストや時間が削減できる。また、データ移行に ETL ツール以外の知識が不要になり、分析者の負担も減る。

ETL ツールを使わず、その都度ユーザがプログラミングし、データの移行をする場合、以下の課題がある。

- データ移行元、データ移行先の DBMS 知識と、プログラミングの知識が必要である。
- 移行するデータ内容の変化によりその都度、プログラム修正を行わなければならない
- 分析環境を変更するときにプログラムも変更しなければならない
- 開発者の異動により解析不能なプログラムが出てきてしまう
- 保守のためにプログラマを半永久的にアサインしなければならない

以上より、ETL ツールはデータ分析を行う上で欠かすことのできないものだと考えられる。

4. 開発するツール概要

現在、IRS に対応した ETL ツールは存在せず、「IRS に大量のデータを蓄積し、必要に応じてデータを他 DB や DWH に移行し、移行先で分析するというシステム」を用いるユーザは、IRS 中のデータをから他 DB や DWH に移行する際のデータ移行を 1 から支援なしに行わなければならず、求められる知識が多いという問題がある。具体的には、分析者は以下の技術を修得している必要がある。

- SQL：IRS からデータを取り出す際には SQL を用いる必要がある。
- Hadoop：HadoopAPI を用いる際には Hadoop の知識と Java によるプログラミングが必要とされる。

これらは一朝一夕に身につけられるものではなく、分析者にとって負担である。また、ビッグデータを活かしたデータ解析は仮説検証型であり、データ分析者は上記のデータ移行を異なるデータ群に対して何度も繰り返す必要がある。その度に IRS と RDB のテーブルの対応関係を考慮し、適切な手段でデータを移行しなければならず、大きな負担になる。

我々はデータ分析者の負担を減らすために、IRS に対応した ETL ツールを作成することとした。

ETL ツールの調査を元に、開発するツールの機能概要を以下に述べる。

- 抽出機能:今回はデータの抽出元は NEC 様の協力で提供していただいた IRS のみを対象とした。
- 変換機能：主にこれまで IRS を用いた分析環境を使用していなかったユーザを対象にしたツールなので、変換については射影や選択などの基本的な機能に限定し、GUI を用いてプログラミングに習熟していない人でも扱いが用意になる点を重視した。
- クレンジング機能：参照整合性制約検査、データ移行先と移行元のデータ型変換などの機能は実装しない。
- ロード機能：データの移行先として解析ツールやデータベースを対象にした。

ウォーターフォール型開発を２回行い、以上の機能を持ったツールを開発する。第 1 イテレーションでは IRS から分析ツールにデータを移行するツールを作成する。その際、データベースのテーブルの射影(表の列を選択すること)、選択(表の行を選択すること)を行うことができる。

第 2 イテレーションでは第一イテレーションでの結果を元に、着手する内容について顧客と相談して決める。

IRS 対応の ETL は今までなかったため、上記の ETL の強みを、IRS を用いた分析環境を使用するユーザに提供することができる。
また、我々の開発するツールは、Hadoop や SQL を使用することができないユーザを特に意識しており、以下の様な特徴を持つ

- ユーザに意識させることがなく、IRS の HadoopAPI を用い高速にデータのロードが可能
- データ移行元、移行先データベースのテーブル関係の定義を GUI で行うことができる。分析者に必須とされている知識や手間を減らし、分析者の負担を軽減することができる。

開発するツールは以下の様な技術的特徴を持つ。

- GUI で転送元、転送先のデータベース間の表・列の対応関係の定義が可能である
- GUI でデータベースの射影(Select)、選択(Where)が可能である
- IRS の HadoopAPI を用いて高速にツールにデータの読み込みが可能である
- ツールに読み込んだデータを一時ファイルを介すことなく、ストリーミング処理で出力先データベースに格納することができる
- データの抽出・加工・挿入はプラグイン化されており、追加の開発が簡単である

5. 利用シナリオ

開発するツールの利用シナリオとして Apache ログデータを分析する状況を以下に提案する。

ある組織では、チームで課題解決のためのシステムを構築する研修”Project Based Learning”(以下、PBL)を約一年かけて行っている。PBL では、顧客に対してヒアリングし、システム設計、実装、テスト、プロジェクト管理を行う。PBL では開発環境として Apache HTTP Server(以下、Apache)と、そのサーバ上で稼働している Redmine や Git を用いている。研修者はチームごとにそれらの環境を用いてプロジェクトを進行する。

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。

指導を行う人間にとって以下の利点があると考えられる。

1. 各チームのアクセス日時データを分析することで、チームごとの特色(休日は一切アクセスしないチーム、夜型のチーム・朝方のチームなど)を明らかにすることができる。これにより、チームの特色に合わせて適切な指

導を行うことができる。
また、期間を区切って分析することで、チームの成長・変化の推移を見出すことができる。

2. 研修者の一定期間ごとアクセスデータを分析することで、研修者個人の役割の変化や仕事量を把握することができる。
3. 分析から得られた発見を蓄積し、来年度の PBL の運営に役立てられる。

また、研修者にとっては以下の利点が考えられる。

1. 長年蓄積されてきたノウハウを参考にプロジェクト運営を行うことができる。
2. 他の研修者と比較し、仕事量や役割など自分の状況を把握しやすくなる。

このような環境において、我々が開発するツールは IRS から分析ツールにデータを移行する際に 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。

本シナリオで Apache ログの蓄積に IRS を用いる理由を以下に述べる。Apache ログは増え続ける一方であり、データベースにはスケーラビリティが求められる。IRS はスケーラビリティに優れており、かつ SQL を用いることができる扱いやすさも兼ね備えており、運用しやすいという特徴を持っているからである。

本シナリオでは、分析ツールとして Pentaho を使用することにした。Pentaho はオープンソースの DWH であり、様々な分析を行うことができる。

今回 Apache ログデータを対象とした利用シナリオを考案した理由は以下の通りである。

1. Apache は世界中でもっとも使われている Web サーバソフトウェアで、大規模な商用サイトから自宅サーバまで幅広く利用されている。
2. Apache のログ解析は専用のソフトウェアがあるほど一般的に行われており、有用性も確認されている。
3. 高度 IT の PBL で Apache を用いており、データの入手が容易である。

Apache ログデータの解析は今回のシナリオ特有のものではなく、広く行われており、本シナリオに則って検証を行えば開発するツールが有用であると判断できると考えた。

利用シナリオの概要図を図 1 に示す。

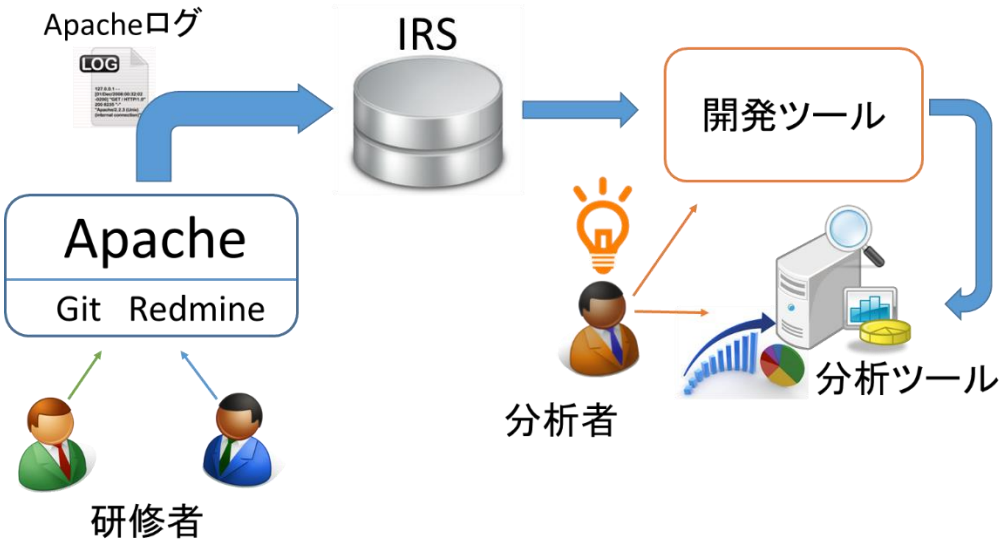


図 1

本シナリオでは、Apache ログは適切な形で IRS に保存してあることを前提としている。

今回のシナリオでは扱うデータを Apache ログに限定したが、IRS や開発ツール、DWH で扱えるデータならば、同様に扱うことができる。

第２項　ステークホルダ

- 顧客側担当者
  日本電気株式会社・システムソフトウェア事業部　白石 雅己様
- 筑波大学担当者
  筑波大学大学院 システム情報系　天笠俊之様
- 想定されるユーザ
  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
- 開発者
  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 コース　斎藤優太 成澤翔平 コバイキ 唐可

第３項　開発体制とリソース

1.　開発体制

開発予定機能毎に責任者を割り当てる方針である。

各イテレーションの要件詳細検討フェーズにて確定させる予定である。

2. 役割

- プロジェクトリーダー：斎藤優太
  - 顧客とチームの窓口
  - ミーティング設定
- 進捗管理：斎藤優太
  - スケジュール立案
  - 進捗管理
- 品質管理：成澤翔平
  - 成果物品質管理
  - プロジェクト環境管理
- マシン管理：コバイキ
  - 開発環境保守運用

3. 開発環境

ツール開発環境

VMWare 上の仮想マシンで開発を行う。
VMware(R) Workstation 10.0.2 build-1744117
OS: CentOS 6.5
メモリ : 2G バイト
プロセッサ : 1 プロセッサ*4 コア
HDD: 40G
Java
Version "1.6.0_30"
OpenJDK Runtime Environment (IcedTea6 1.13.3) (rhel-5.1.13.3.el6_5-x86_64)
OpenJDK 64-Bit Server VM (build 23.25-b01, mixed mode)

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

Pentaho
Kettle - Spoon version GA リリース - 5.0.1-stable

Build date　　: 2013-11-15_16-08-58

IRS 構築環境

IRS の構築環境を表 1 と図 2 に示す。

各ノードは以下のマシンを使用している。1 ノード 1 マシンを割り当てている。

OS: CentOS6.5

CPU:Intel Corei7-3770

メモリ　15.5G バイト

表 1

		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

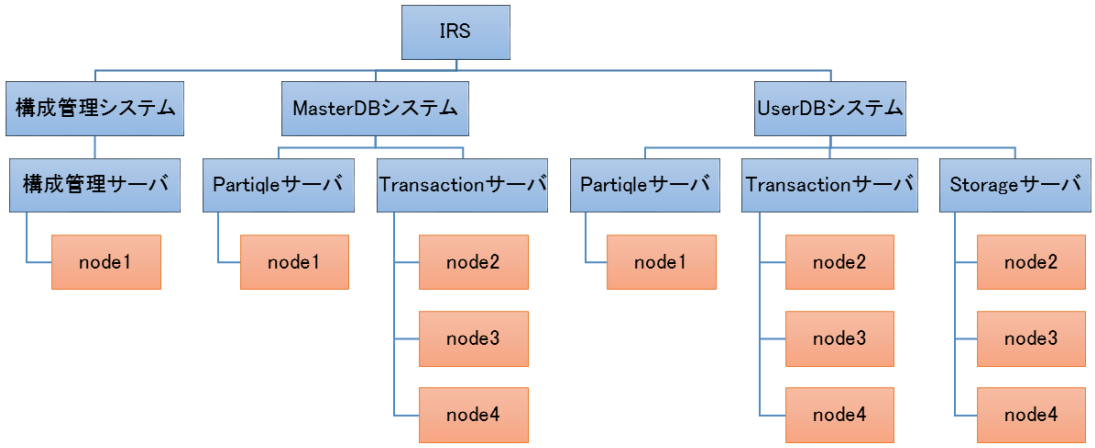


図 2

第 4 項　スケジュール

スケジュール概要

プロジェクト終了予定は 12 月上旬までに 2 回のウォーターフォール型開発を行う。これは、開発チームのシステム開発経験が浅く、また IRS に関するツール開発

に携わるのが初めてのため、本プロジェクトを一回のウォーターフォール型開発で行うには進捗や見積についてリスクが存在すると判断したためである。よって、ウォーターフォールを 2 回行い、機能を分割して開発することで顧客と合意した。

第 1 回目のウォーターフォール型開発を第 1 イテレーションと呼ぶ。2 回目のウォーターフォール型開発を第 2 イテレーションと呼ぶ。

第 1 イテレーションの前に全体の要件を決め、開発構想書を作成する期間を設けた。この期間に大まかな機能要件とスケジュール、開発環境の構築を行う。その後、第 1 イテレーションを開始し、設計・開発を行う。第 2 イテレーションでは、顧客から第 1 イテレーションの成果物のレビューを頂き要件や優先順位の見直しを行う。変更後の要件と優先順位について顧客と合意し第 2 イテレーションを開始する。第 2 イテレーション修了後、特定課題研究報告書の執筆や発表用デモンストレーションの作成を行う期間を設けた。

開発準備期間(2014/05/15~ 2014/06/11)

本イテレーションでは開発構想書の作成を行う。開発構想書の内容は開発体制・開発内容、開発スケジュールの概要である。開発構想書について顧客と合意することを本イテレーションの終了条件とする。

また、IRS や開発環境の構築についても本イテレーションで行う。

第 1 イテレーション(6/11~10/14)

第 1 イテレーションは表 3.3-1 のスケジュールで行う。成果物については第 4 章成果物を参照してください。

開発構想書の開発概要に基づき要件を明確化する、また優先順位をつけ、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・開発に着手する。

成果物の受け入れ条件や第 1 イテレーションの終了条件については要件詳細検討フェーズにて顧客と合意する。

表 2

フェーズ名	期間	成果物
要件詳細検討・外部設計	6/11~7/16	要件定義書、用語集、ユースケース記述, 画面設計書、システム周辺図
内部設計	7/16~7/30	クラス図、シーケンス図、プログラム詳細仕様書
実装 1	7/30~8/3	なし
夏休み	8/4~8/18	なし
実装 2	8/19~10/6	なし
テスト	10/6~10/9	テスト計画書、テスト報告書、テストケース一覧

納品準備	10/9~10/14	ユーザマニュアル、セットアップガイド、検収検査書、知財関連書類、納品書

第２イテレーション(11/10~12/10)

第２イテレーションのスケジュールについては、期間は 11/10~12/10 とし、第１イテレーションで作成した NET ツールの拡張や品質の向上や、解決しきれていないバグを取りのぞく。また、第１イテレーションで実装した機能の改善と、新機能として Transform 機能のプラグイン化を行う。
第２イテレーションの詳細は第 2 章第 1 節第 6 項の第２イテレーション詳細にて述べる。

報告準備期間(12/01~12/31)

特定課題研究報告書の執筆や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。

第５項　第１イテレーション成果物

- 開発準備フェーズ
  - 開発構想書：開発の背景や目的、開発内容の概要、開発体制、スケジュール概要などについて顧客と合意するために作成する。以降の開発は開発構想書を元に詳細を確定させながら進行する。

- 要件定義詳細検討フェーズ
  - 要件定義書：開発構想書で合意した開発内容を更に詳細に顧客と合意し記述する。該当イテレーションで開発する範囲を明確にするために作成する。
  - ユースケース記述：要件定義で提案した機能の振る舞いについて詳細を決定し定義する。顧客の持つツール機能像とチームのツール機能像のすり合わせを行うために作成する。

- 外部設計フェーズ
  - 画面設計書：GUI の画面上のボタンや操作について顧客と合意し、その内容を記述する。顧客の持つツール像とチームのツール像のすり合わせを行うために作成する。
  - システム周辺図：我々の開発するツールと IRS や他システムの関係を記述する。顧客の持つツール像とチームのツール像のすり合わせ

を行うために作成する。

- 内部設計フェーズ
  - クラス図:プログラムのクラスの関係を記述する。各担当の責任範囲とクラスの影響範囲を明確化するために作成する。
  - シーケンス図:機能の内部的な進行過程を記述する。他システムや、他機能との関係するタイミングを明確化するために作成する。
  - プログラム詳細仕様書：プログラムのクラスや関数について仕様や引数の説明を記述する。コーディング規約などのルールを記述しメンバーで共有するために作成する。
- 実装フェーズ
  - 特になし
- テストフェーズ
  - テスト計画書:テスト計画を記述する。この計画に基づきテストを実施する。テスト実施を円滑にするために作成する。
  - テスト報告書:実施したテストについて環境や実施概要、結果を記述する。顧客にテスト結果を報告するために作成する。
  - テストケース一覧：テスト計画に基づき作成したテストケースの一覧を記述する。品質の高いテストを行うために作成する。
- 納品準備フェーズ
  - 納品書:納品物についての説明を記述する。納品物に漏れがないかを確認するために作成する。
  - ユーザマニュアル：NET ツールについてユーザの使い方を記述する。
  - セットアップガイド:NET ツールのセットアップについて手順を記述する。
  - 検収検査書:顧客の受け入れ条件を満たし、受け入れて頂けたかを記述する。

第６項　第２イテレーション詳細

第２イテレーションでは、チームの円滑なコミュニケーションを取るためにコアタイムを導入する。コアタイムの時間は以下のとおりである。

月	火	水	木	金
10~12 時	10~12 時	10~12 時	10~12 時	10~12 時
13~15 時		13~15 時		13~15 時

第２イテレーションのスケジュールの詳細を表３にのせる。

表 3

フェーズ名	期間	成果物
要件詳細検討・外部設計	11/10~11/14	調査報告書 要件定義書
実装	11/17~11/21	なし
テスト	11/25~12/1	テスト報告書、テストケース一覧
納品準備	12/1~12/3	納品報告書 ユーザマニュアル Plugin セットアップガイド Eclipse でのビルド手順書

第７項　第２イテレーション成果物

- 調査フェーズ
  - 調査報告書：顧客と打ち合わせた大まかな機能の要求について実現可能性の調査、詳細な要件決定のための調査を行い、結果を報告する。

- 要件定義フェーズ
  - 要件定義書：開発構想書で合意した開発内容を更に詳細に顧客と合意し記述する。該当イテレーションで開発する範囲を明確にするために作成する。
  - 変更記録：第１イテレーションで行った要件定義と第２イテレーションで行った要件定義に差がある場合に、その差について報告する。

- 設計フェーズ
  - 画面設計書：GUI の画面上のボタンや操作について顧客と合意し、その内容を記述する。顧客の持つツール像とチームのツール像のすり合わせを行うために作成する。
  - ユースケース記述：要件定義で提案した機能の振る舞いについて詳細を決定し定義する。顧客の持つツール機能像とチームのツール機能像のすり合わせを行うために作成する。
  - **型変換マニュアル**：Transform 機能で行う型変換のルールについて記述する。(ユーザマニュアルの一部として作成する)

- 内部設計フェーズ
  - Javadoc：プログラムのクラスや関数について仕様や引数の説明を記述する。
- 実装フェーズ
  - 特になし

- テストフェーズ
  - 単体テストテストケース一覧：行った単体テストの概要を一覧にして報告する。
  - 機能テスト報告書:要件定義、ユースケース記述に基づき行った機能テストの結果について報告する。
  - 総合テスト報告書：ユーザマニュアルに基づき行った総合テストの結果について報告する。
- 納品準備フェーズ
  - 納品書:納品物についての説明を記述する。納品物に漏れがないかを確認するために作成する。
  - ユーザマニュアル：NET ツールについてユーザの使い方を記述する。
  - Plugin セットアップガイド：NET ツールのセットアップについて手順を記述する。
  - **Eclipse でのビルド手順書**：NET ツールのソースコードからビルドするまでの手順を説明する。
  - **Trnasform プラグイン作成マニュアル**：NET ツールに Transform 機能を追加する場合に、Transform プラグインの作成手順を説明する。

第８項　第２イテレーションプロジェクト運営方針

Redmine

- Redmine の使用理由：前回では、開発支援ツールが活用できず、進捗管理や工数管理がプロジェクトで積極的に管理できていなかった。
  今回は Redmine の使用ルールを明確に定め、メンバーの進捗や仕事量を管理する。
- Redmine のチケットの発行ルールを以下に示す。

トラッカー
・バグ：バグを発見した時に使用する。
・機能：成果物が発生する作業を行うときに使用する。
・サポート：成果物が発生しない作業を行うときに使用する。

説明
・機能名：機能名を書く
・種別：バグフィックス、新機能、実装を見送った機能の実装　のどれか
・機能概要：機能概要を書く
・不安要素：不安要素や要調査項目があれば書く
・顧客との打ち合わせ：顧客との打ち合わせが　必要　か　不要　かを書く
・本チケットで管理する作業の概要

ステータス
・新規：チケットだけ作成して、未着手の場合
・進行中：調査中など、作業を行っている場合
・解決：作業を終了したが、チームミーティングで承認を受けていない場合
・終了：作業についてチームミーティングで承認を受け、作業を終了する場合
・問題発生：発生した問題についてチームで共有した場合
(他のステータスは使わない)

優先度
使わない

担当者
担当者を選ぶ

期日
チケットが「終了」になる予定日を書く

予定工数
基本的に一日 5 時間の工数を考えているので、それと期日を考慮して書く

進捗率
使用しない

Git

- Git の使用理由：第 1 イテレーションでは Git の使用にルールを設けなかったため、Redmine と連携ができておらず、メンバーの仕事の進捗や量の管理ができていなかった。よって、今回は Redmine と Git の連携ルールを定め、定期ミーティングで確認する。
- Git の使用ルールを以下に示す
  - Git にコミットする前に対応するチケットを作成・確認する
  - Git のコミットメッセージに #チケット番号 という記述を一番初めに入れ、作業内容を記述する。

第９項　成果物の公開

本プロジェクトでの成果物の一部を一般に公開する予定である。公開方法は未定である。

公開することにより、以下のユーザに対して価値が発生すると考えられる。

- NET ツールを利用する人 : NET ツールにより、データ移行作業が支援され、負担が軽減される。
- データ移行プログラムを作成する人：NET ツールのソースコードには、IRS の HadoopAPI や IRS JDBC を用いたプログラミングが含まれているため、IRS からデータベースへのデータ移行の際に参考にすることができる。

第３章 要件定義フェーズ

本章では、要件定義フェーズでの成果物について述べる。成果物は、機能一覧と非機能要件定義書である。

第１節　第１イテレーション機能一覧

第 1 節機能一覧では、プロジェクト開始時に作成し合意した第１イテレーションで実現する機能の一覧の内容を記述する。第１イテレーション機能詳細一覧からの引用である。

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節　第１イテレーション非機能要件定義書

第 2 節では、プロジェクト開始時に作成し合意した非機能要件の内容を記述する。非機能要件検討資料からの引用である。

非機能要件名	基準
可用性	特に基準は設けない
性能	特に基準は設けないが、目標として以下の性能を達成するように努力する ・ロード性能は、PostgreSQL にシングルスレッドでインサートするよりも早いこと ・ツールを使用したデータ移行にかかる時間は、IRS からファイルを経由してデータ移行する場合より短いこと
拡張性	● 第１イテレーションではプラグイン化は意識しない
運用	● ユーザマニュアルを作成する。
保守性	● セットアップマニュアルを作成する。 ● 関数の詳細は Javadoc に記述する
移行性	● 本ツールの基本動作環境は CentOS とする。 ● CentOS 依存のツールにならないように注意を払い開発する。
セキュリティ	● 本ツールは悪意のないユーザを対象とする。 ● ユーザから入力された情報は、暗号化するなど適切に扱うことに注意する。
システム環境	特に基準は設けない
エコロジー	特に基準は設けない

第３節　第１イテレーションユースケース図・記述

第 3 節では、プロジェクト開始時に作成し合意したユースケース図及びユースケース記述の内容を記述する。

第１項　ユースケース図

作成したユースケース図を図 3 に示す。

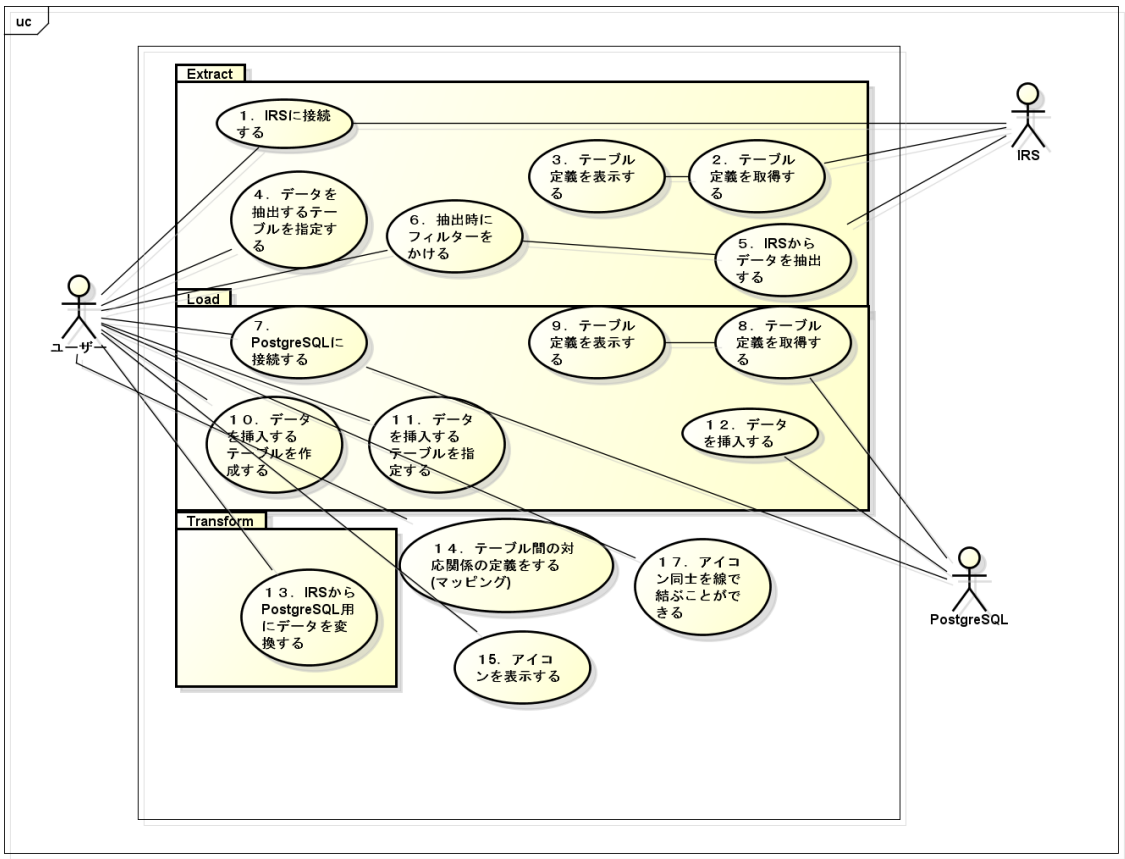

uc
Extract
Load
Transform
１．IRSに接続する
２．テーブル定義を取得する
３．テーブル定義を表示する
４．データを抽出するテーブルを指定する
５．IRSからデータを抽出する
６．抽出時にフィルターをかける
７．PostgreSQLに接続する
８．テーブル定義を取得する
９．テーブル定義を表示する
１０．データを挿入するテーブルを作成する
１１．データを挿入するテーブルを指定する
１２．データを挿入する
１３．IRSからPostgreSQL用にデータを変換する
１４．テーブル間の対応関係の定義をする(マッピング)
１５．アイコンを表示する
１７．アイコン同士を線で結ぶことができる
ユーザー
IRS
PostgreSQL

図 3

第２項　ユースケース記述

ユースケース記述を以下に示す。

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を取得する
3	IRS から取得したテーブル定義データを表示する
4	データを抽出するテーブルを指定する
5	PostgreSQL に接続する
6	PostgreSQL からテーブル定義データを取得する
7	PostgreSQL から取得したテーブル定義データを表示する
8	PostgreSQL にデータを挿入するテーブルを作成する
9	データを挿入するテーブルを指定する
10	抽出元テーブルから抽出するカラムを選択する
11	テーブル間の対応関係の定義を行う(マッピング)
12	プロジェクトを実行する

機能番号	1
機能名	IRS に接続する
目的	IRS からテーブル定義を入手する、IRS からデータを抽出するために IRS に接続する。
事前条件	● ツールが起動している ● プロジェクトが作成されている
基本系列	1. ユーザが、データソースウィンドウの「inputDB」を右クリックする 2. ユーザが、入力元の DB として「IRS」を追加する 3. ユーザが、追加した IRS の「接続情報」をダブルクリックする 4. ユーザが、接続情報ウィンドウに以下の内容を入力する ➢ HadoopAPI 接続時に使用するストレージサーバの URL ➢ HadoopAPI 接続時に使用するトランザクションサーバの URL ➢ 接続先 Partiqle サーバホスト名(IP アドレス) ➢ トランザクションサーバの URL ➢ ユーザ名 ➢ パスワード

	➢ 接続先 Partiqle サーバポート番号 5. ユーザが、「connect」ボタンを押す
代替系列	4.1 入力した内容に入力抜けや不備があり、接続が確立できない場合 4.1.1 入力された情報に基づき、IRS に接続を試みる 4.1.2「接続に失敗した」旨のメッセージを表示する
事後条件	4. IRS との接続が確立されている。

機能番号	2
機能名	IRS からテーブル定義データを取得する
目的	IRS からテーブル定義データと例として先頭のデータを取得する
事前条件	1. IRS との接続が確立されている 2. 機能番号 1 の 5.「connect」ボタンを押す　の直後である
基本系列	1. IRS との接続が確立されると、ツールが、テーブル定義情報を自動で取得する 2. ツールが、データ例としてテーブル内のデータを 1 行取得する 3. ツールが、「テーブル定義情報が取得された」旨のメッセージを出力する 4. ツールが、接続情報ウィンドウを表示する
代替系列	1.1 テーブル定義情報が取得できない場合 1.1.1「テーブル定義情報の取得に失敗した」旨のメッセージを表示する 1.1.2 接続情報ウィンドウを表示する 1.1.3「get table」ボタンが押せるようになる 2.1 テーブルの中身が空の場合 2.1.1 データ例の場所に「テーブルの中身が空である」ことを示す
事後条件	1. テーブル定義情報が取得されている 2. データ例が取得されている 3. 「get table」ボタンが押されるようになっている
備考	「get table」ボタンを押すと、テーブル情報を再取得する。 テーブル定義情報の更新を行うことができる。

機能番号	3

機能名	IRS から取得したテーブル定義データを表示する
目的	テーブル定義情報とデータ例をユーザに示すため
事前条件	1. テーブル定義情報が取得されている 2. データ例が取得されている
基本系列	1. ツールが、取得したテーブルをデータソースウィンドウに表示する 2. ユーザが、テーブルをダブルクリックする 3. ツールが、テーブル情報の詳細を表示する 4. 表示される情報は以下の通りである ➢ 主キーかどうか ➢ カラム名 ➢ カラム型 ➢ カラム型の桁数 ➢ NULL 値を許容するかどうか ➢ データ例 5. ユーザが、「閉じる」ボタンを押す 6. ツールが、テーブル情報詳細ウィンドウを消す
代替系列	なし
事後条件	1. テーブル定義情報がデータソースウィンドウに表示されている
備考	以下の方は特別な表示を行う ・decimal 型：「カラム型の桁数」に「精度」と「位取り」をカンマ区切りで表示する。

機能番号	4
機能名	データを抽出するテーブルを指定する
目的	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
事前条件	1. テーブル定義情報がデータソースウィンドウに表示されている。
基本系列	1. ユーザが、データを抽出するテーブルを決定する 2. ユーザが、決定したテーブルをデータソースウィンドウから作業ウィンドウにドラック&ドロップする 3. ツールが、作業ウィンドウに選択したテーブルのアイコンを表示する
代替系列	なし
事後条件	1. 作業ウィンドウにテーブルのアイコンが表示されている

機能番号	5
機能名	PostgreSQL に接続する
目的	PostgreSQL からテーブル定義を入手する、PostgreSQL からデータを抽出するために PostgreSQL に接続する。
事前条件	1. ツールが起動している 2. プロジェクトが作成されている
基本系列	1. ユーザが、データソースウィンドウの「outputDB」を右クリックする 2. ユーザが、出力先の DB として「PostgreSQL」を追加する 3. ユーザが、追加した「PostgreSQL」をダブルクリックする 4. ユーザが、接続情報ウィンドウに以下の内容を入力する ➢ サーバのホスト名 ➢ サーバのポート(デフォルト：5432) ➢ データベース名 ➢ ユーザネーム ➢ パスワード 5. ユーザが、「connect」ボタンを押す
代替系列	4.1 入力した内容に入力抜けや不備があり、接続が確立できない場合 4.1.1 入力された情報に基づき、PostgreSQL に接続を試みる 4.1.2「接続に失敗した」旨のメッセージを表示する
事後条件	1. PostgreSQL との接続が確立されている。

機能番号	6
機能名	PostgreSQL からテーブル定義データを取得する
目的	PostgreSQL からテーブル定義データを表示するため
事前条件	1. PostgreSQL との接続が確立されている 2. 機能番号 7 の 5.「connect」ボタンを押す　の直後である。
基本系列	1. PostgreSQL との接続が確立されると、ツールが、テーブル定義情報を自動で取得する 2. ツールが、データ例としてテーブル内のデータを 1 行取得する 3. ツールが、「テーブル定義情報が取得された」旨のメッセージを出力する 4. ツールが、接続情報ウィンドウを表示する

代替系列	1.1 テーブル定義情報が取得できない場合 1.1.1「テーブル定義情報の取得に失敗した」旨のメッセージを表示する 1.1.2 接続情報ウィンドウを表示する 1.1.3「Get table」ボタンが押せるようになる 2.1 テーブルの中身が空の場合 2.1.1 データ例の場所に「テーブルの中身が空である」ことを示す
事後条件	4. テーブル定義情報が取得されている 5. データ例が取得されている

機能番号	7
機能名	PostgreSQL から取得したテーブル定義データを表示する
目的	テーブル定義情報をユーザに示すため
事前条件	1. テーブル定義情報が取得されている 2. データ例が取得されている
基本系列	1. ツールが、テーブル定義情報をデータソースウィンドウに表示する 2. ユーザが、テーブルをダブルクリックする 3. ツールが、テーブル情報の詳細を表示する 4. 表示される情報は以下の通りである ➢ 主キーかどうか ➢ カラム名 ➢ カラム型 ➢ カラム型の桁数 ➢ NULL 値を許容するかどうか 5. ユーザが、「閉じる」ボタンを押す 6. ツールが、テーブル情報詳細ウィンドウを消す
代替系列	なし
事後条件	1. テーブル定義情報がデータソースウィンドウに表示されている

機能番号	8
機能名	PostgreSQL にデータを挿入するテーブルを作成する
目的	PostgreSQL にデータ移行先としてテーブルを作成する

事前条件	1. PostgreSQL との接続が確立されている
基本系列	1. ユーザが、「Create Table」アイコンをデータソースウィンドウから作業ウィンドウにドラック&ドロップする 2. ツールが、「Create Table」アイコンを作業ウィンドウに表示する 3. ユーザが、「Create Table」アイコンをダブルクリックする 4. ツールが、テーブル作成ウィンドウを表示する 5. ユーザが、テーブル作成ウィンドウに表示されている以下の項目を入力する ➢ テーブル名 ➢ カラム名 ➢ カラム型 ➢ 主キー指定 ➢ NOT NULL 制約 ➢ UNIQUE 制約 6. ユーザが、「追加」ボタンを押し、カラムを追加する 7. ユーザが、「テーブル作成」ボタンを押す 8. ツールが、テーブル作成ウィンドウを消す
代替系列	7.1 追加したカラムを削除する場合 7.1.1 カラムをクリックで選択し、下部の「削除」ボタンを押すことでカラムは削除される 7.2 カラムが 0 のテーブルを作成しようとした場合 7.2.1「カラムが追加されていない」旨のメッセージを表示する 7.2.2 テーブル作成ウィンドウを表示する
事後条件	1. テーブルアイコンが作業ウィンドウに表示される
備考	実際にテーブル作成が行われるのは、「プロジェクトの実行」ボタンが押された後である。 それまでは、「テーブルが作成されている」という想定のもとツールは動作する。

機能番号	9
機能名	データを挿入するテーブルを指定する
目的	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
事前条件	1. テーブル定義情報がデータソースウィンドウに表示されている。
基本系列	1. ユーザが、データを抽出するテーブルを決定する 2. ユーザが、決定したテーブルをデータソースウィンドウから作業ウィ

	ンドウにドラック&ドロップする 3. ツールが、作業ウィンドウに選択したテーブルのアイコンを表示する
代替系列	なし
事後条件	1. 作業ウィンドウにテーブルのアイコンが表示されている

機能番号	10
機能名	抽出元テーブルから抽出するカラムを選択する
目的	GUI を用いて抽出するカラムを選択する。
事前条件	1. 「データ移行元テーブル」アイコンと「データ移行先テーブル」アイコン、「Extract」アイコン、「Transform」アイコンが作業ウィンドウに表示されている。 2. それぞれのアイコンが適切な形で連結されている。
基本系列	1. ユーザが、「Extract」アイコンをファンクションウィンドウから作業ウィンドウにドラッグ&ドロップする 2. ツールが、作業ウィンドウに「Extract」アイコンを表示する 3. ユーザが、「データ移行元テーブル」アイコンと「データ移行先テーブル」アイコンと「Extract」アイコンを線で結ぶ 4. ユーザが、「Extract」アイコンをダブルクリックする 5. ツールが、作業ウィンドウに新しく「Extract」タブを作成する 6. ユーザが、「Extract」タブをクリックする 7. ユーザが、カラムにチェックを付けることで抽出するカラムを選択する 8. 抽出条件を設ける場合は、ユーザが、「抽出条件」をクリックする 9. ツールが、「条件指定」ウィンドウを表示する 10. ユーザが、抽出条件、抽出範囲を入力する 11. ユーザが、「OK」ボタンを押す
代替系列	8.1 詳細記述欄を用いて条件を指定する場合 8.1.1 詳細記述欄を用いる場合、表中の抽出条件は使用できない 8.1.2 ユーザが、詳細記述欄に Between や AND などを用いて条件を記述する。 11.1 抽出対象のカラムを 1 つも選ばない場合 11.1.1「抽出対象のカラムが 1 つもされていない」旨のメッセージを表示する 7.1.2「Extract」タブを表示する

事後条件	1. 各テーブルのカラム間の対応付けが完了している
備考	表中の抽出条件で入力できる情報は以下の通りである。このうち一つだけ指定することができる。 A：抽出元カラム X：ユーザ入力 Y：ユーザ入力 ➢ BETWEEN：X ≦A ≦Y ➢ IN：X,Y,… ➢ 比較演算子(<,>,=,≦,≧,≠)：例)A ＜ X 詳細記述欄では、上記の抽出条件を AND,OR,NOT で複数指定することができる

機能番号	11
機能名	テーブル間の対応関係の定義を行う(マッピング)
目的	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
事前条件	3. 「データ移行元テーブル」アイコンと「データ移行先テーブル」アイコンが作業ウィンドウに表示されている
基本系列	1. ユーザが、「Transform」アイコンをファンクションウィンドウから作業ウィンドウにドラッグ&ドロップする 2. ツールが、作業ウィンドウに「Transform」アイコンを表示する 3. ユーザが、「データ移行元テーブル」アイコンと「データ移行先テーブル」アイコンと「Transform」アイコンを線で結ぶ 4. ユーザが、「Transform」アイコンをダブルクリックする 5. ツールが、作業ウィンドウに新しく「Transform」タブを作成する 6. ユーザが、「Transform」タブをクリックする 7. ユーザが、各テーブルのカラムを線で紐づけることでデータ移行の対応付けを行う
代替系列	7.1 一つのカラムも対応付けしなかった場合 7.1.1「対応付けが 1 つもされていない」旨のメッセージを表示する 7.1.2「Transform」タブを表示する
事後条件	2. 各テーブルのカラム間の対応付けが完了している

機能番号	12
機能名	プロジェクトを実行する
目的	データ移行を開始するため
事前条件	1.　データ移行プロジェクトが作成されている 2.　データ移行プロジェクトが矛盾なく、実行できる
基本系列	1.　ユーザが、「プロジェクトの実行」ボタンを押す 2.　ツールが、データ移行を実行する 3.　ツールが、「データ移行が正常に終了した」旨のメッセージを表示する
代替系列	2.1 データ移行がエラーで異常終了した場合 2.1.1 ユーザにエラーメッセージを表示し、プロジェクトを終了する。
事後条件	1.　ツールが起動している
備考	2. データ移行が行われる　の詳細は第 1 イテレーション機能詳細一覧を参照してください。 関連する機能番号は、5, 6, 12, 13, 15。 二度目の異常終了後の移行先テーブルの内容は異常終了したままの状態である。

第４節　　第２イテレーション要件定義

第２イテレーションで実装する機能の調査・要件定義・外部設計を本節で述べる。
ユースケース記述については、第１イテレーションとの差分やユーザの視点から考えて、9. ユーザが入力した抽出条件を扱う機能　についてのみ記述した。

第２イテレーション実装予定機能一覧

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

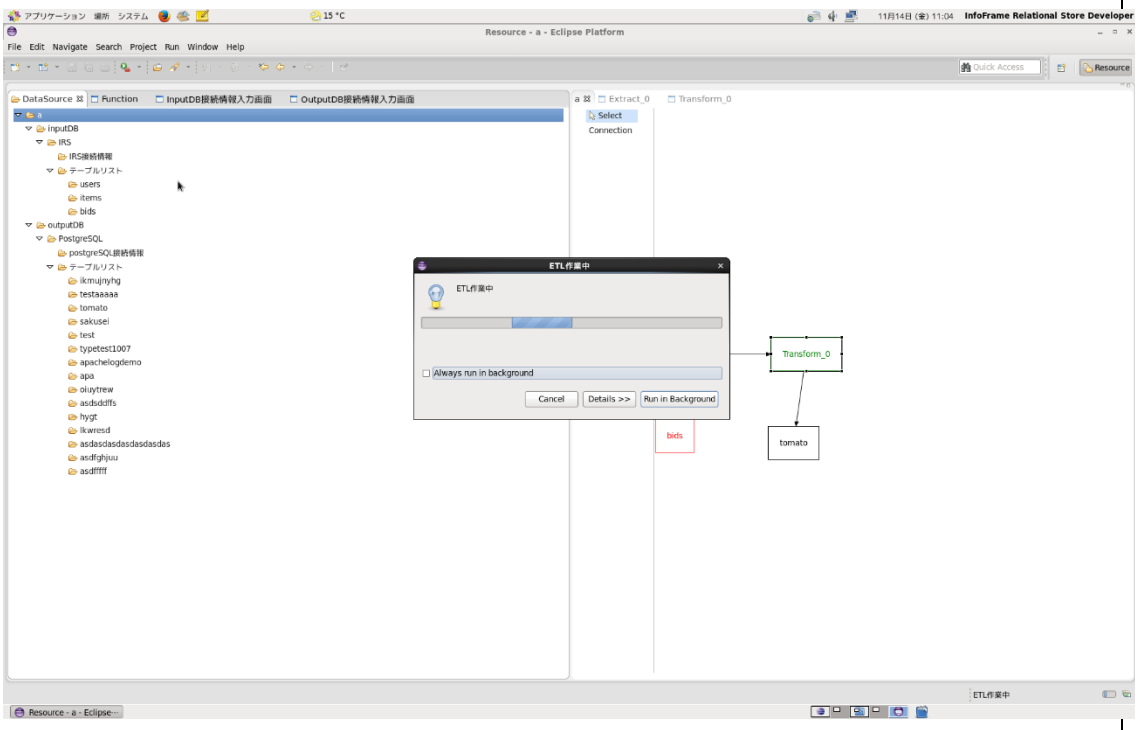


図 4

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

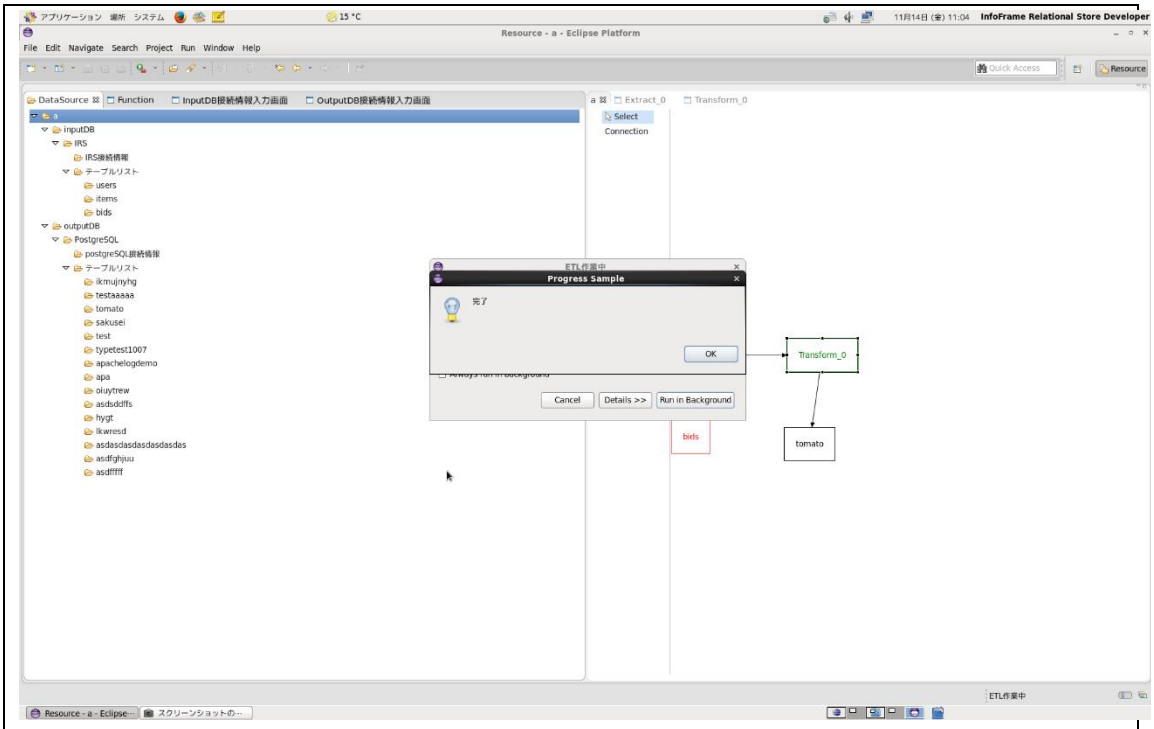


図 5

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	あり

影響	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**

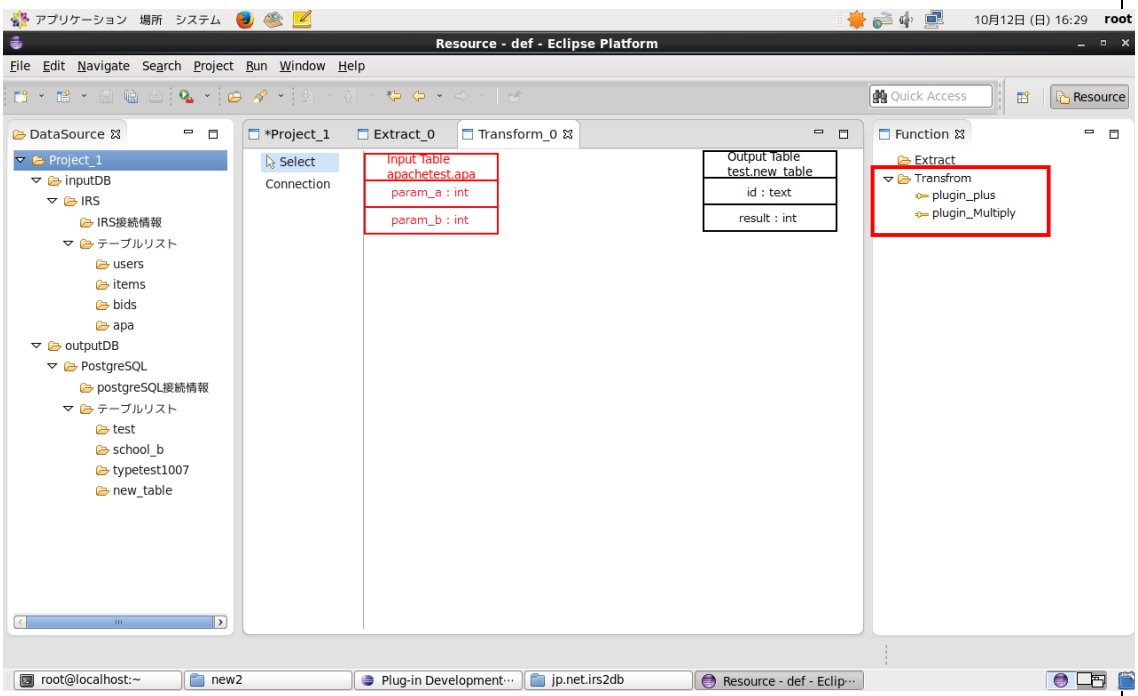
  プラグイン作成マニュアル　を作成する。

備考

プラグイン作成マニュアルを作成する。

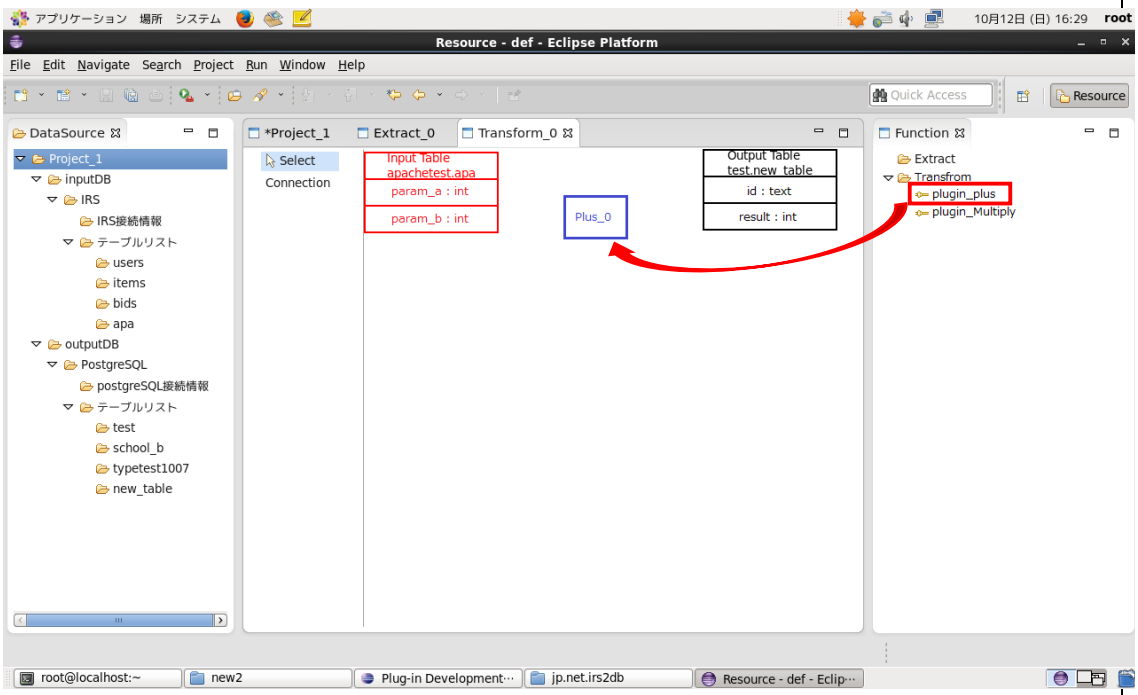
画面イメージ

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。

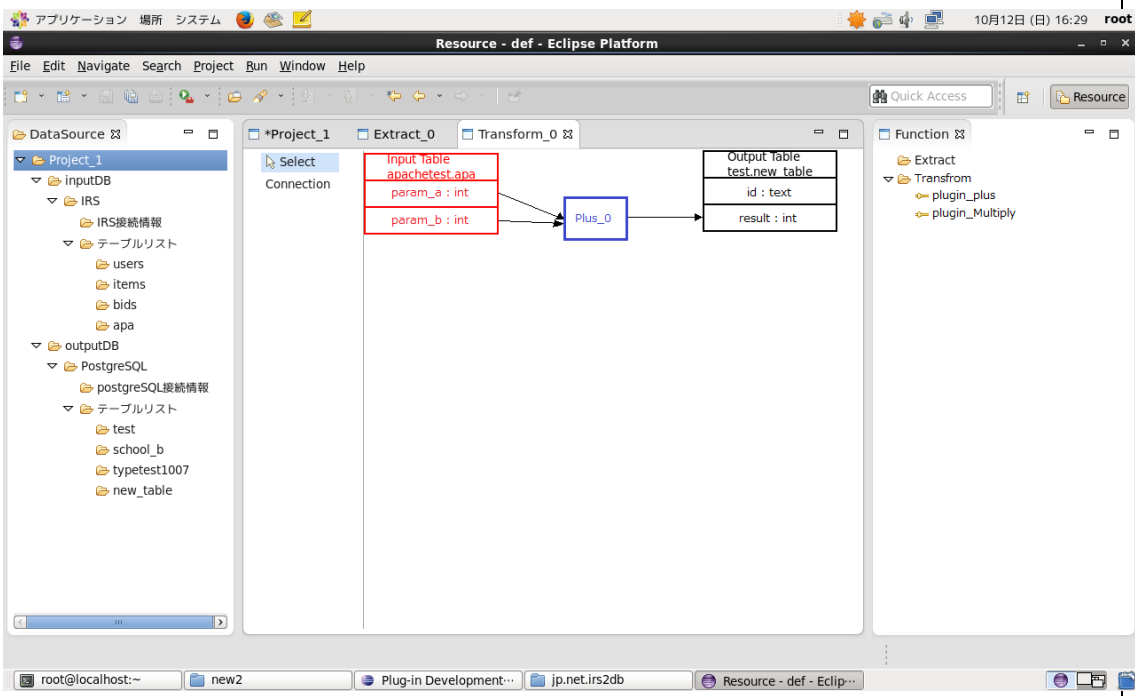


2. 使用するプラグインを選択し、Transform 編集ウィンドウにドラッグアンドドロッ

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。

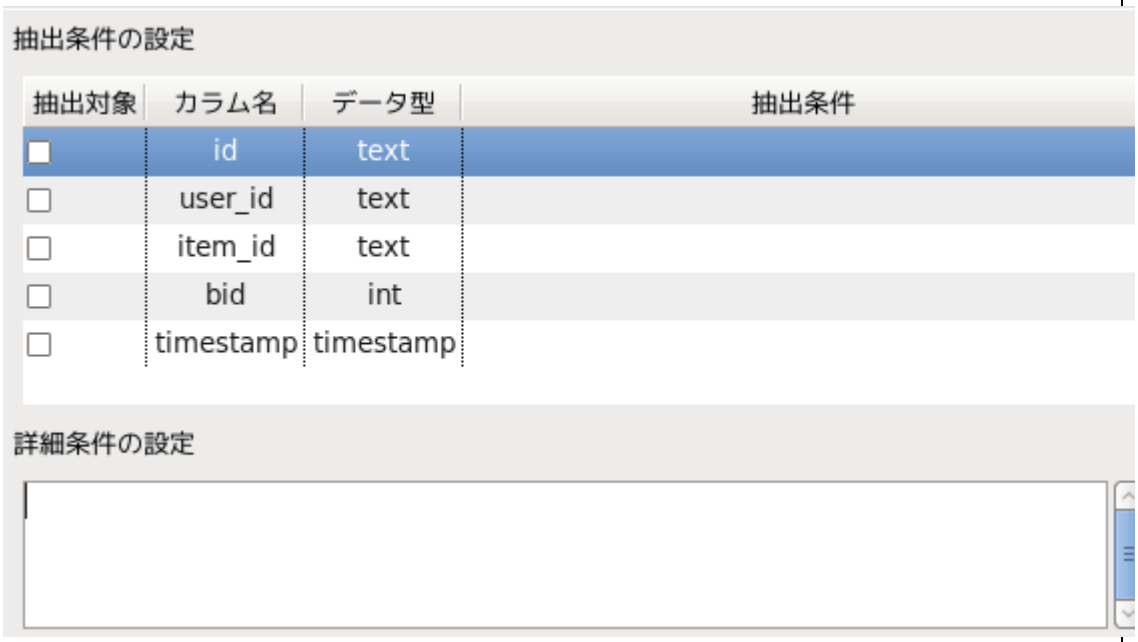


図 6

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 7

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

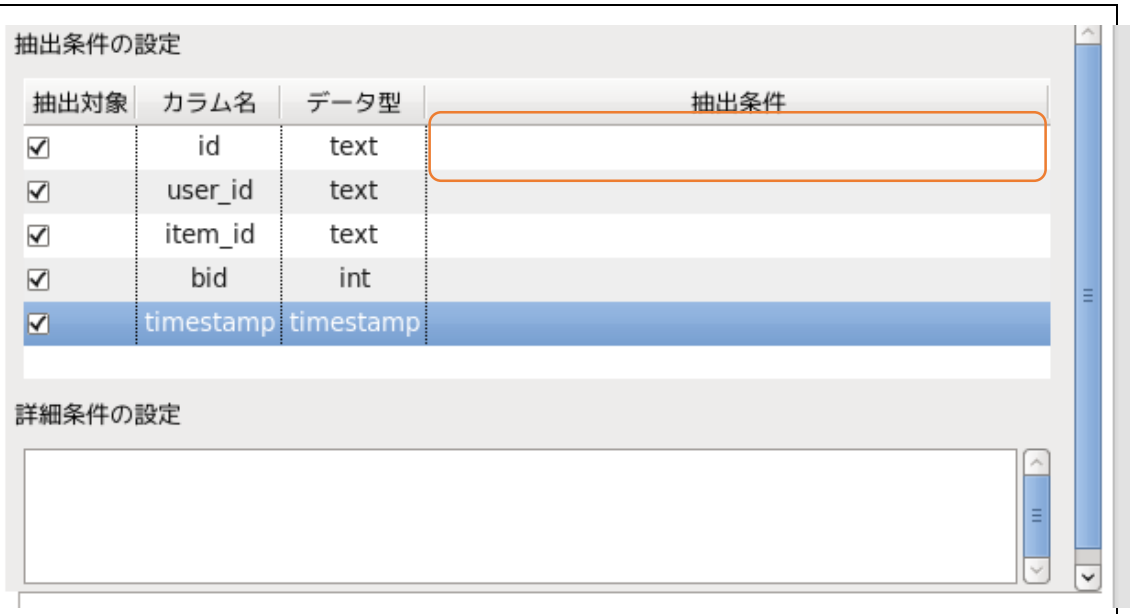


図１

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。

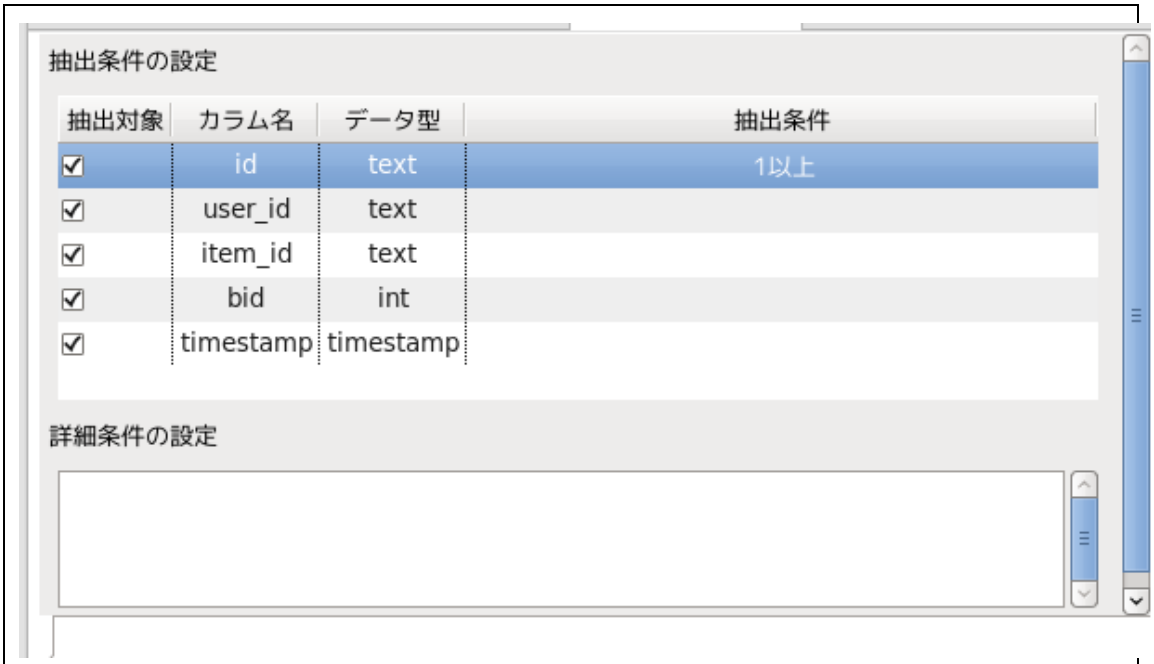


図３

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

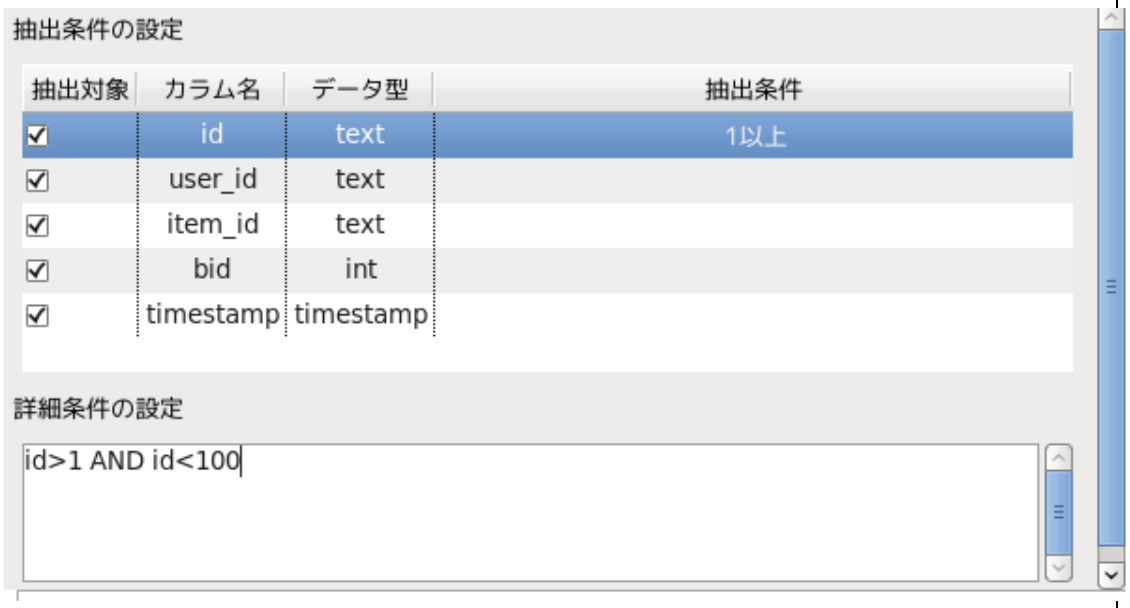


図４

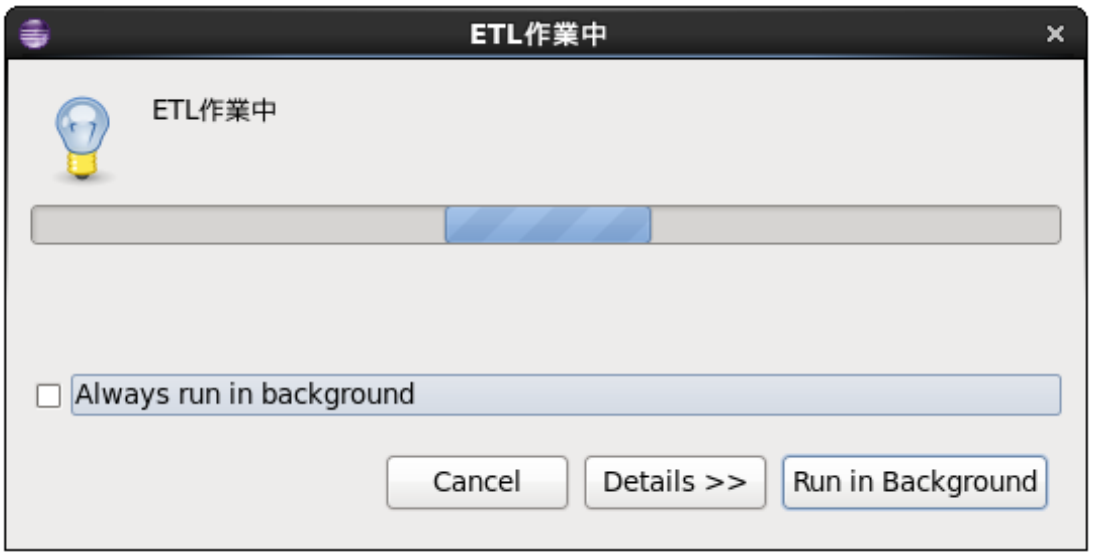
詳細条件の設定のテキストボックスに抽出条件を入力する。この場合、カラムごとの抽出条件がされていてもカラムごとの抽出条件は無視され、図４のように入力されたテキストが抽出条件として使用される。

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

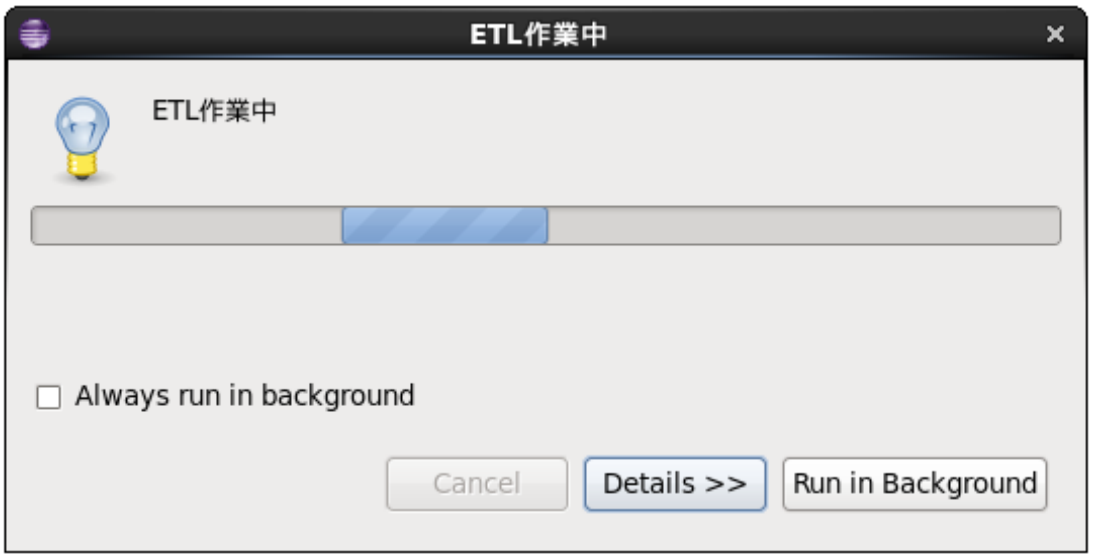
ETL 作業中では以下のようなポップアップが表示される。

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。

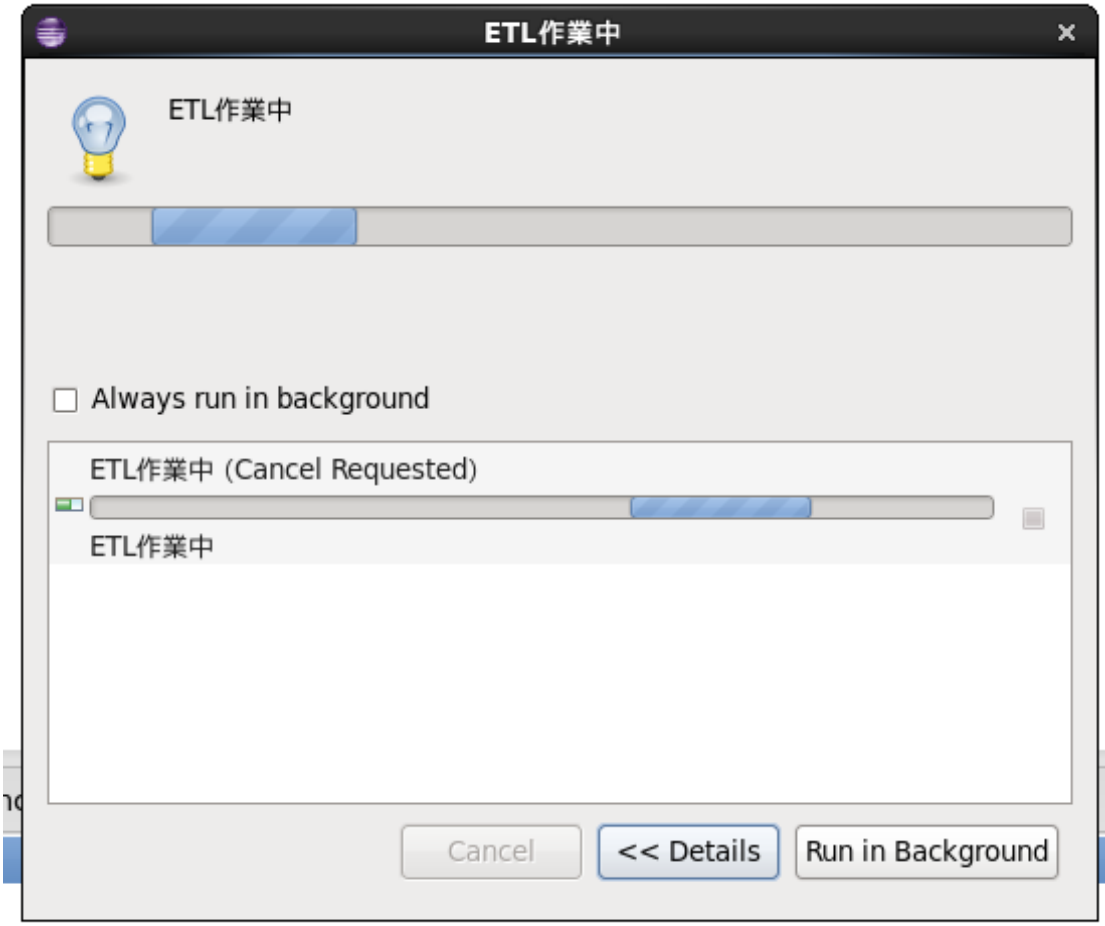


このキャンセルボタンを押すと ETL 作業を中止する。

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。

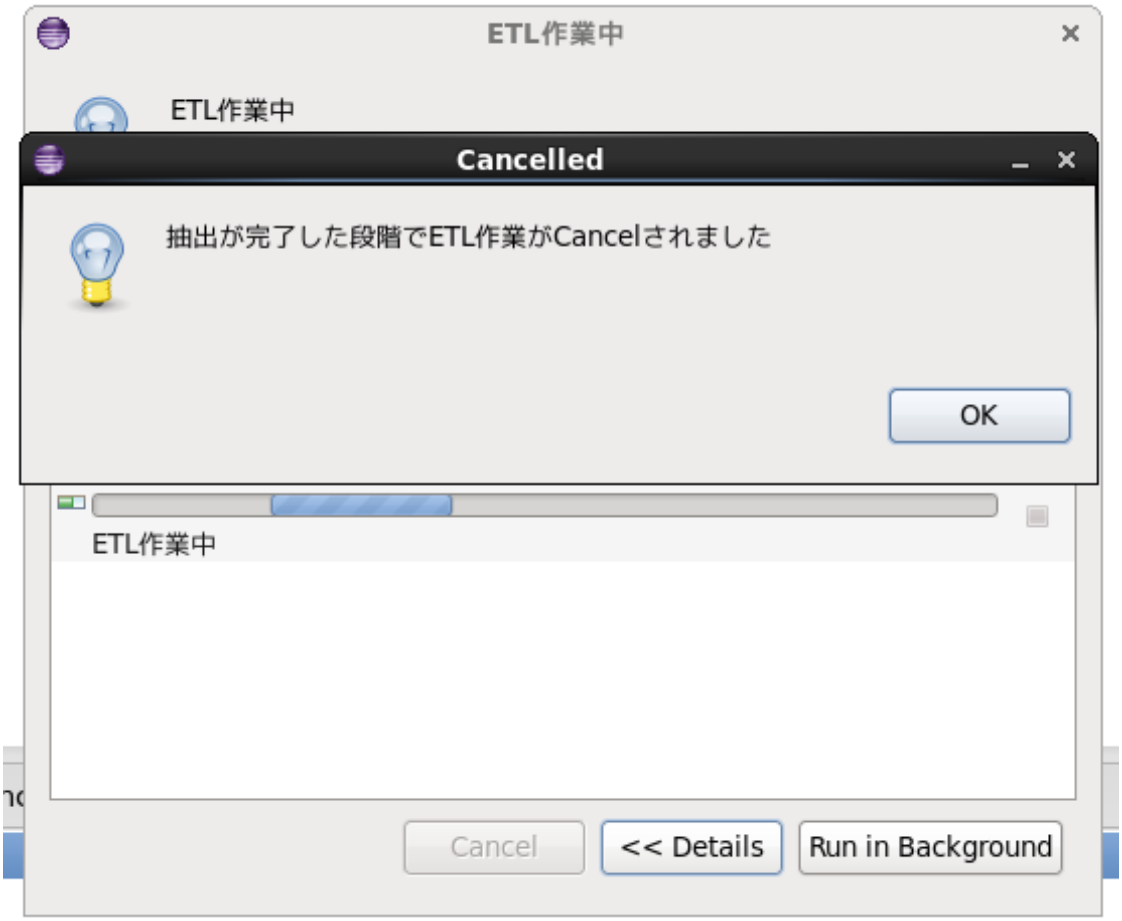


この状態で Detail ボタンを押すと以下のように ETL 作業が中断準備中であることが表示される。



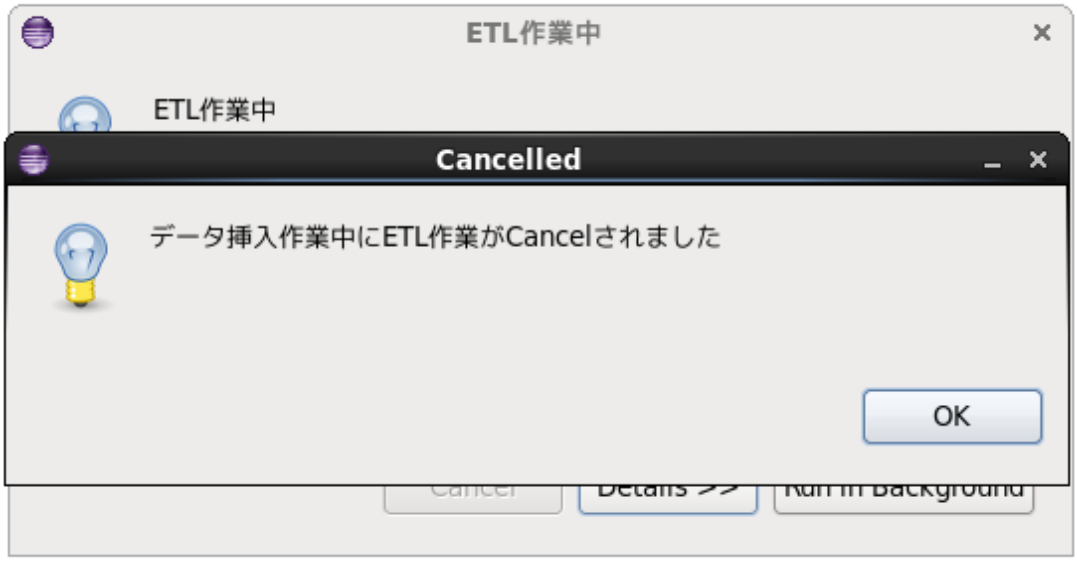
中断するタイミングは２つあり、

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。

第４章 第１イテレーション外部設計フェーズ

本章では外部設計フェーズで作成した成果物について述べる。成果物は GUI 検討書類である。

第１節　GUI 検討書類

本節では GUI について合意した内容を記述する。

第１項　GUI ストーリー概要

本項では想定したツール使用のストーリーを述べる。ストーリーは「DB 間のテーブルのデータ移行」を想定した。

1. 入力、出力先となる DB に接続する。
   (ア)成功した場合
      ①　テーブルの自動取得を開始する
   (イ)失敗した場合
      ①　ユーザにエラー表示
2. 接続した DB からテーブル定義を取得する。
   (ア)成功した場合
      ①　テーブル定義を追加する
   (イ)失敗した場合

① ユーザにエラー表示

3. テーブル定義を参照した上で、移行するテーブルを選択する。
4. 選択したテーブルを作業ウィンドウにドラック＆ドラップする
   (ア)既存出力先テーブルを選択する場合
      ① 出力先 DB からドラック＆ドロップする
   (イ)新規出力先テーブルを作成する場合
      ① テーブル一覧の最下段の Create New Table をドラック&ドロップ
      ② 作業ウィンドウ内の Create New Table をダブルクリックする
      ③ UI に従って作成する
5. 抽出、加工関数を作業ウィンドウにドラック＆ドラップする。
6. 関数と入力、出力先のテーブルを紐づける。
7. 抽出関数をダブルクリックし、入力データベースから抽出する条件を設定する。
   (ア)カラム間の紐づけの中に Between などの機能を入れることで抽出条件を設定することができる。
   (イ)紐づけを解除することで抽出する/しないを選択できる。
8. 作業ウィンドウに戻り、加工関数をダブルクリックする。
   (ア)入力テーブルと出力テーブルのカラム間の対応付けを紐づけすることで行う。
   (イ)カラム間の紐づけの中に Multiply など機能入れることでデータの加工を行うことができるようになる予定である。(第１イレテーションでは予定外である。)
9. 5~9 で対応関係が取れた場合はプロジェクトを実行する。
   (ア)成功した場合
      ① 出力先となる DB にデータが移行され、ユーザに完了したことを通知する。
   (イ)失敗した場合
      ① プロジェクトが不完全
         1. 不完全な理由をエラーメッセージとしてユーザに提示する
      ② 実行中に接続が不安定もしくは切断された
         1. 作業が中断されたことを通知し、これまでの処理は残して作業を終了する。

第２項　GUI 詳細

本項では第１項のストーリーの基づき操作する場合の GUI の画面を解説する。

まず、project を作成し、その中の inputDB を右クリックして、入力元の DB として IRS を追加する。

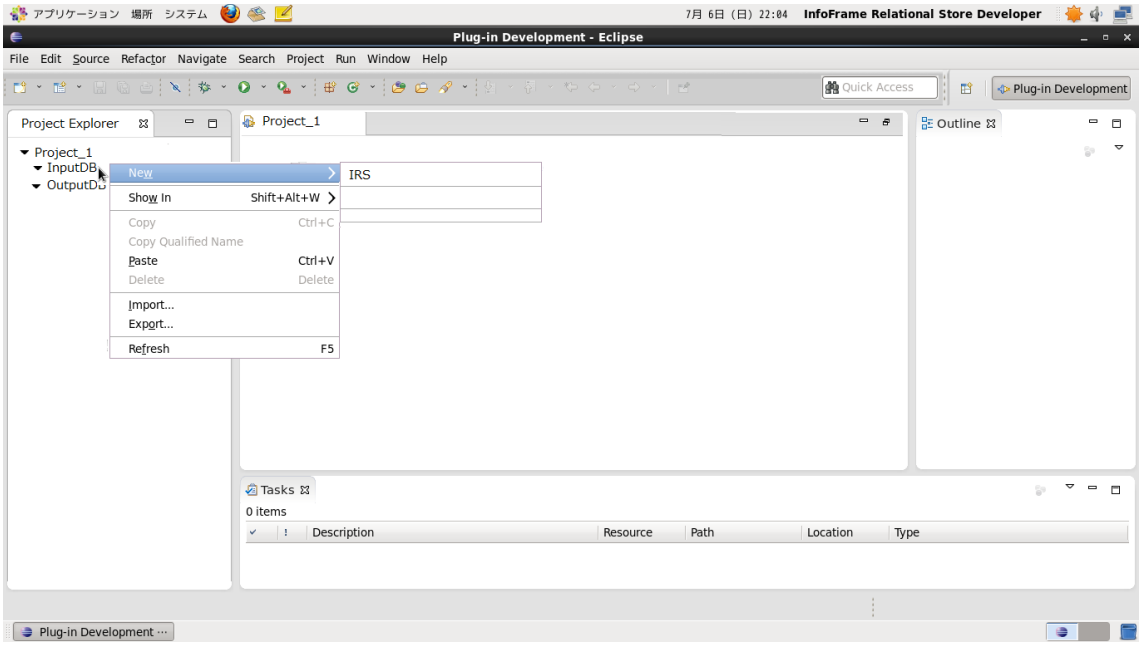


図 8 右クリックで DB を追加する

outputDB に関しても同様に追加する。操作を行った結果が図 5 である。

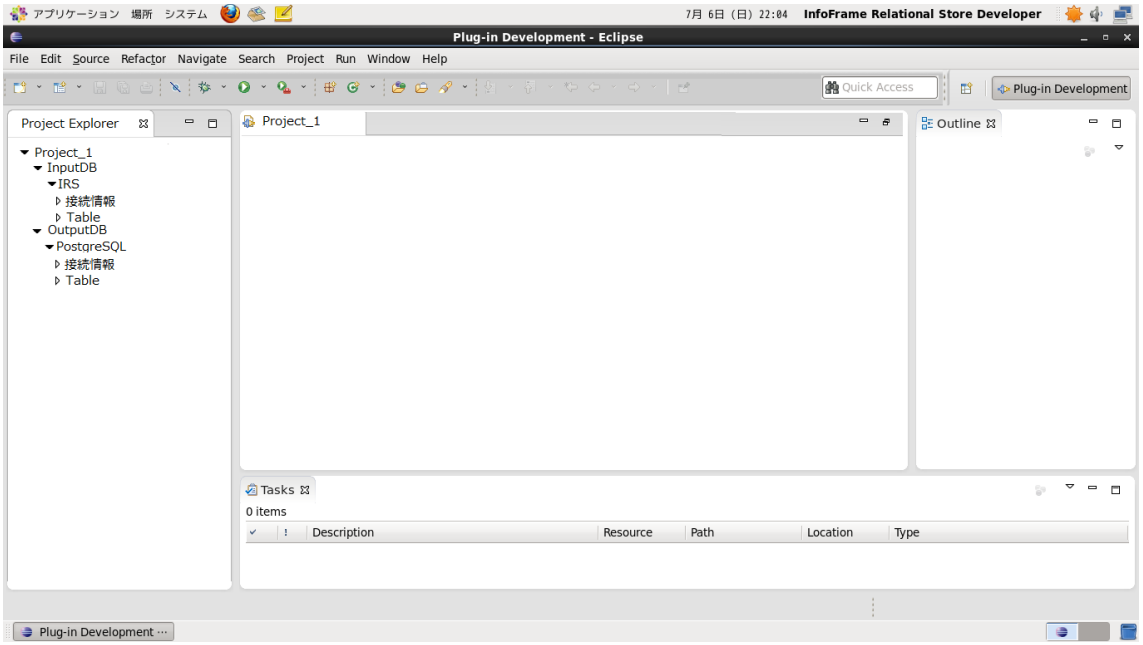


図 9 DB の追加結果

次に追加した DB に接続するために「接続情報」をダブルクリックする。ダブルクリックすると、作業ウィンドウのタブに新たに接続情報ウィンドウが追加される。

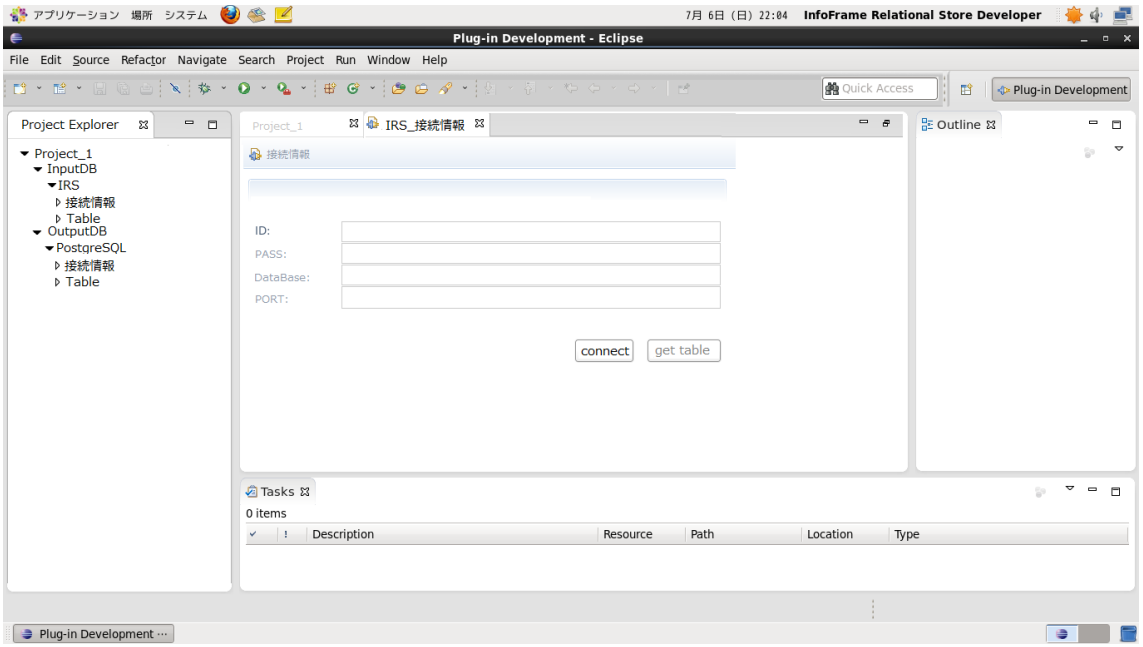


図 10　接続情報入力ウィンドウ

ID と PASS、DB などを入力し、「connect」ボタンを押すと指定の DB に接続を試みる。接続に成功すると、図 7 のような画面が表示され、テーブル情報を自動で取得する。

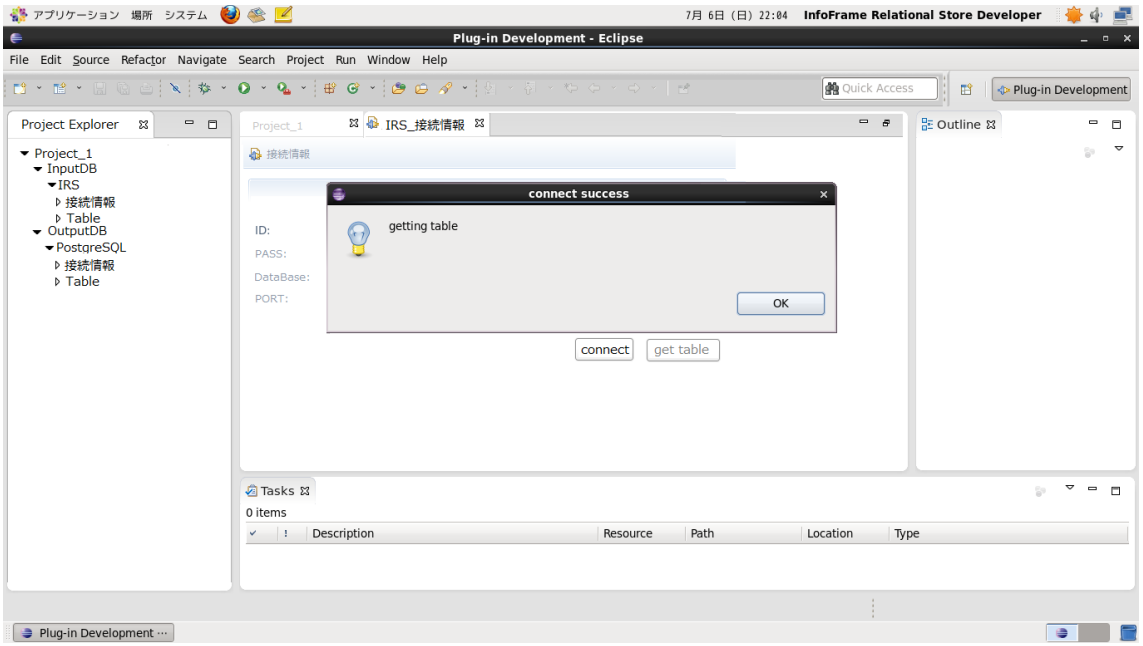


図 11　DB への接続の成功

接続に失敗すると、図 8 のようにエラーメッセージが表示される。

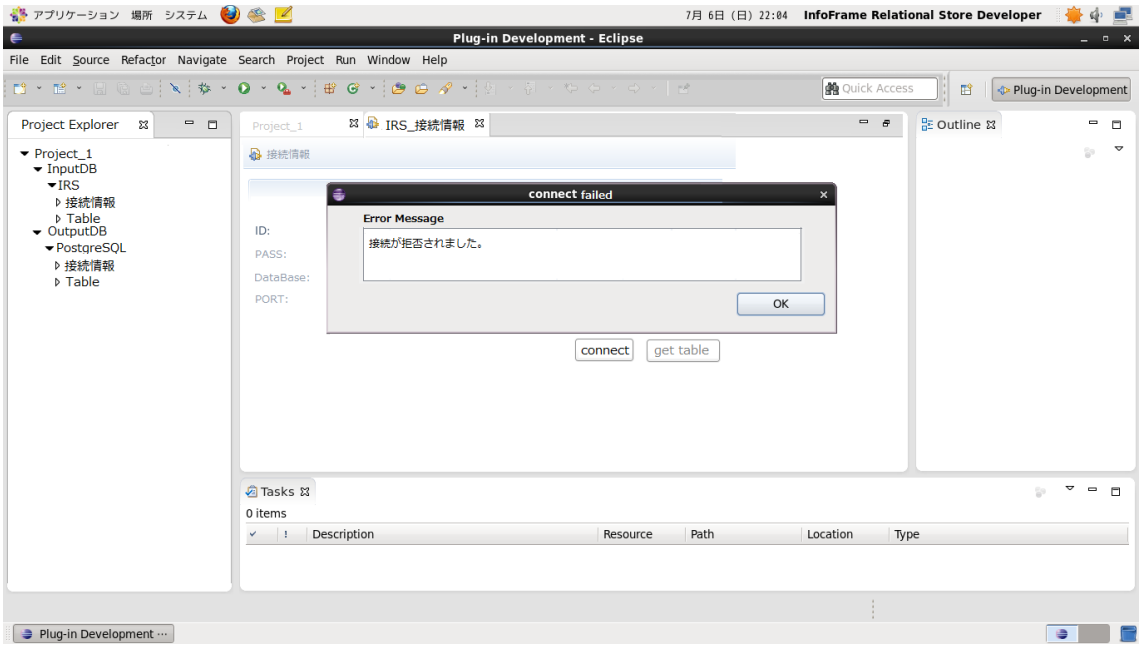


図 12 DB への接続の失敗

テーブルの取得に成功すると、図 9 のようにテーブル情報が追加される。

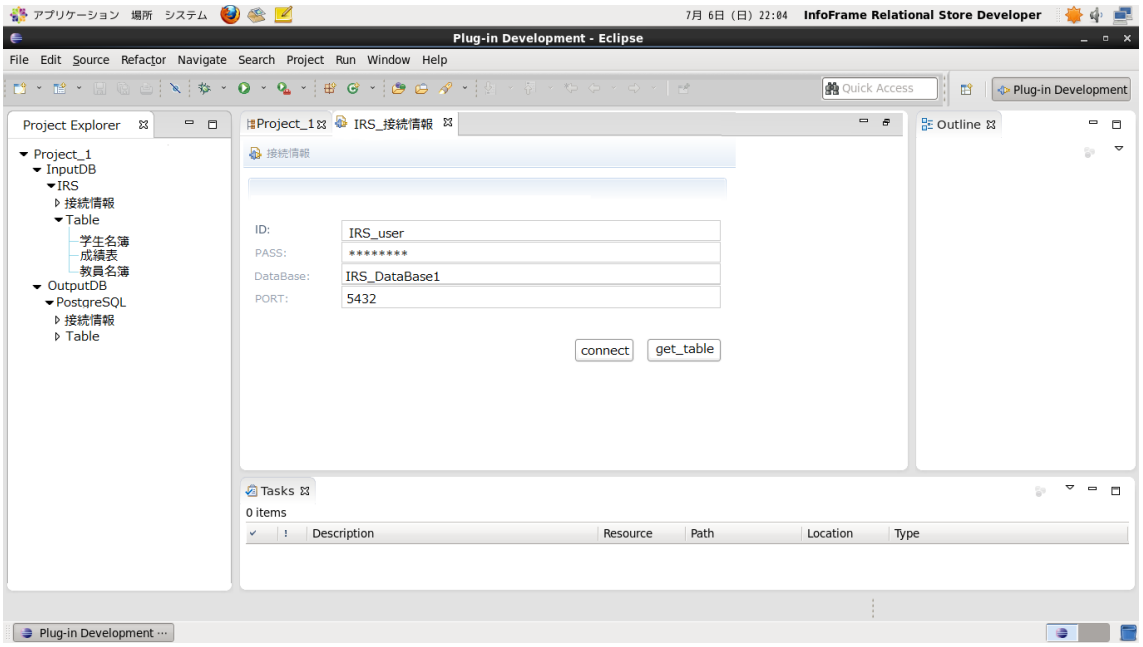


図 13　テーブルの取得に成功

失敗した場合には図 10 のようにエラーメッセージが表示される。

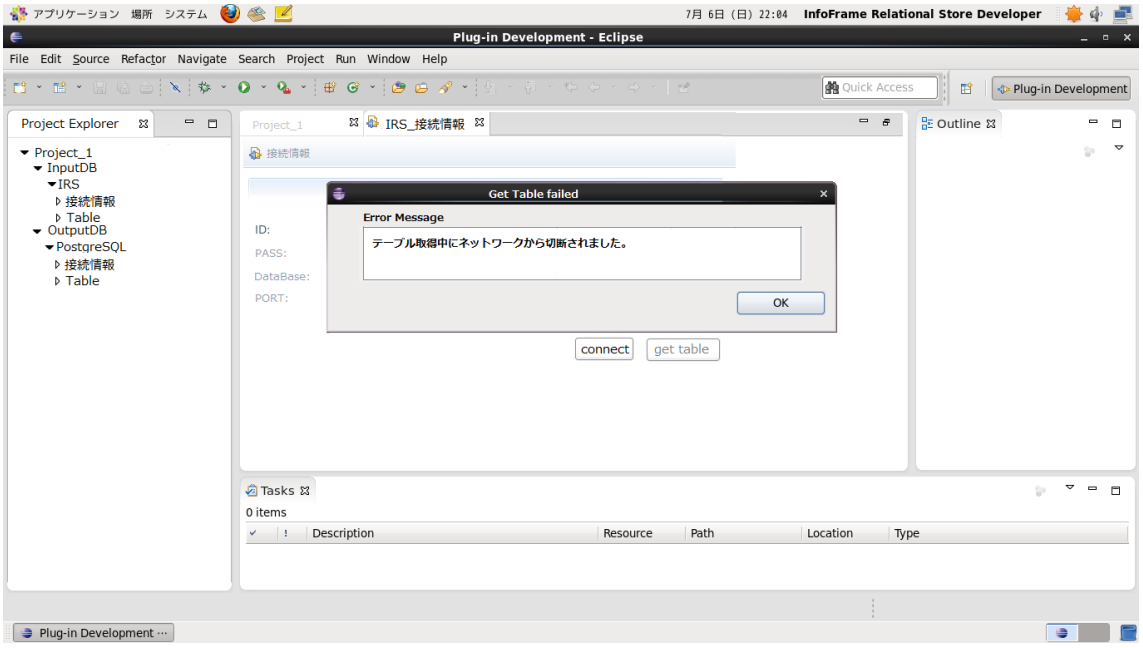


図 14 テーブルの取得に失敗

また、テーブル定義を確認するには、確認したいテーブルをダブルクリックすることで図11のような画面が表示され、確認することができる。

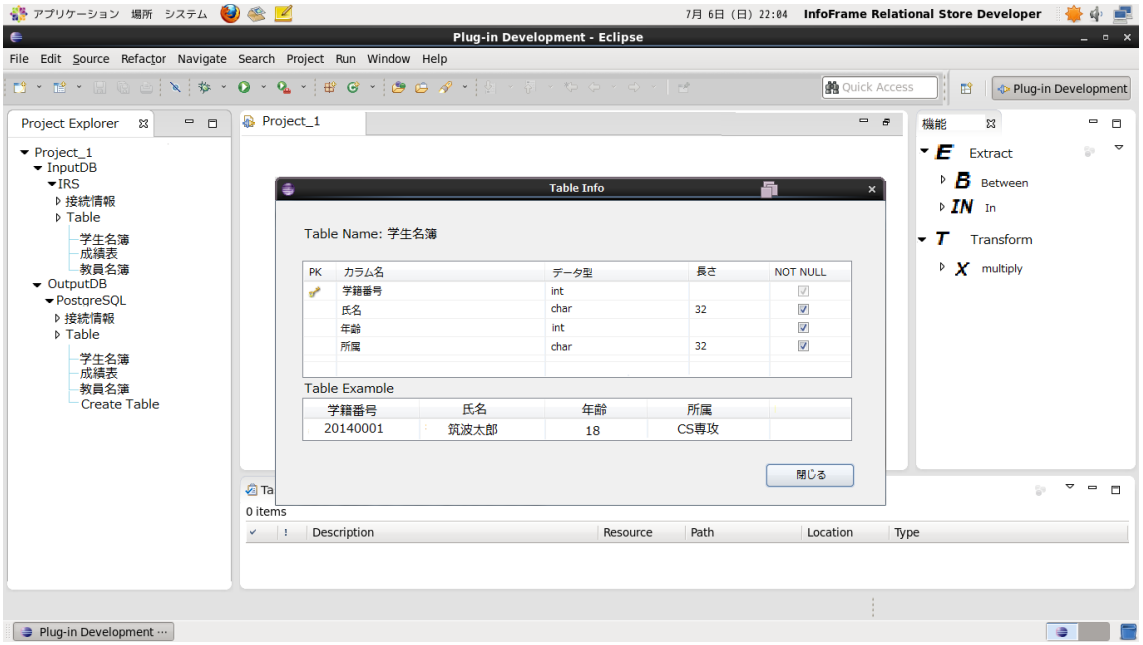


図 15　テーブルの確認

そして、データの移行したい既存テーブルを input、output から作業ウィンドウにドラック&ドロップし、機能ウィンドウからも Extract(抽出)、Transform(加工)関数をドラック＆ドロップする（図 12）。アイコンについては暫定的なものである。

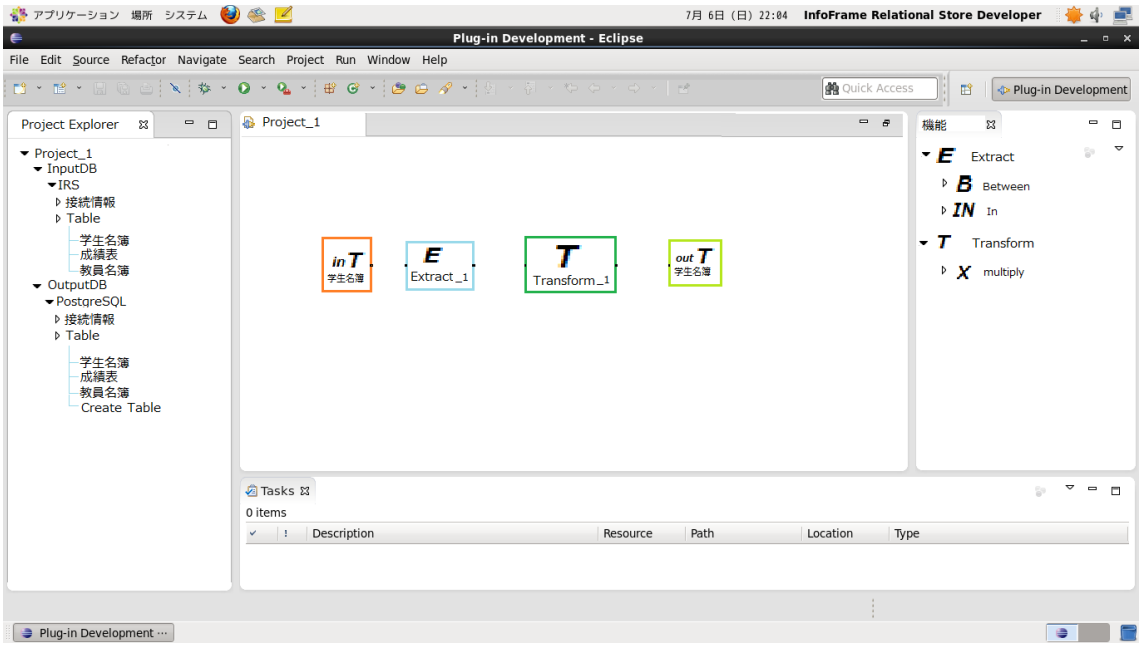


図 16　作業ウィンドウの画面例

この際に、OutputDB に新規のテーブルを作成する場合には OutputDB の Create New Table を作業ウィンドウにドラック&ドロップし、アイコンをダブルクリックすると、図 13 のような画面が表示され、必要な値を入力し、作成することができる（図 14，15）。

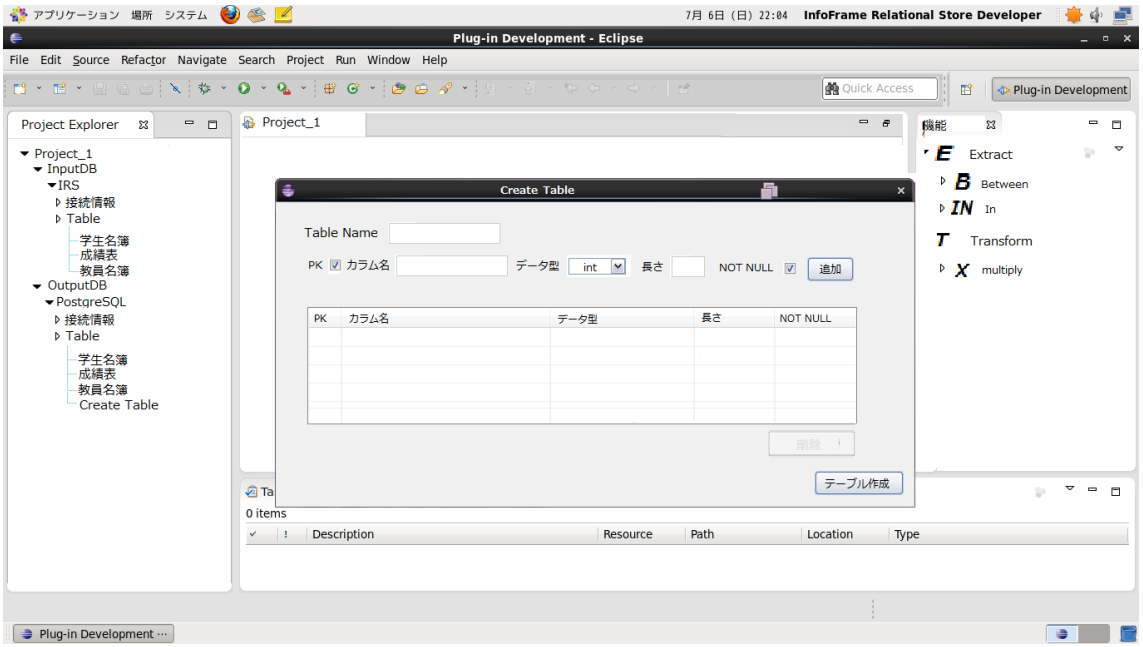


図 17 新規出力先テーブルの作成画面 1

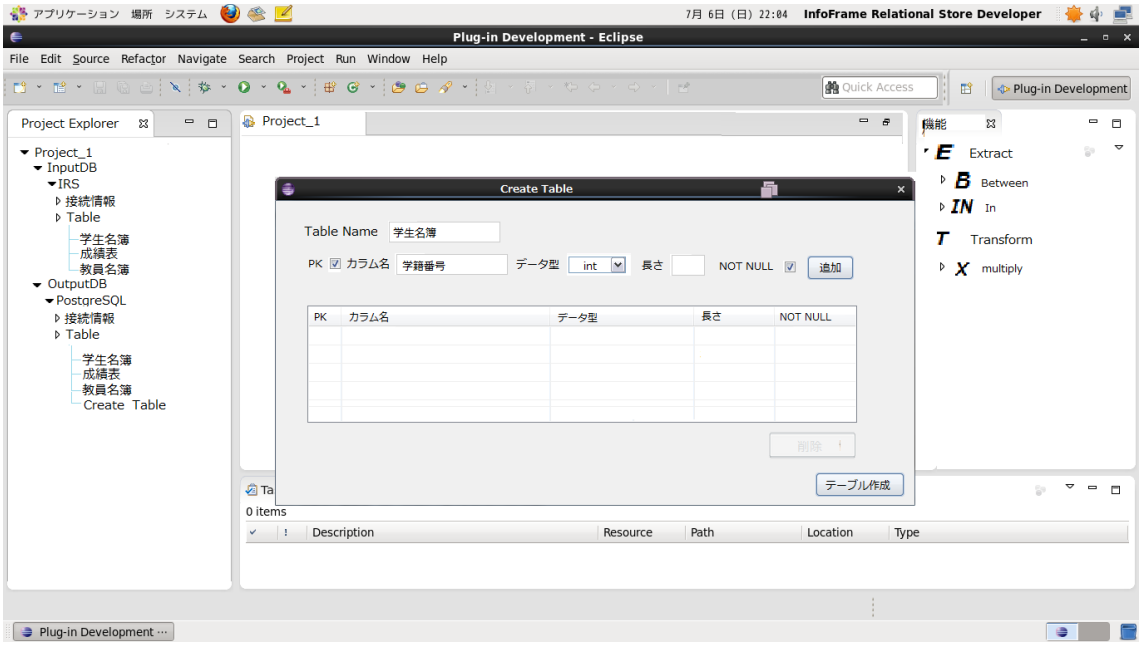


図 18 新規出力先テーブルの作成 2

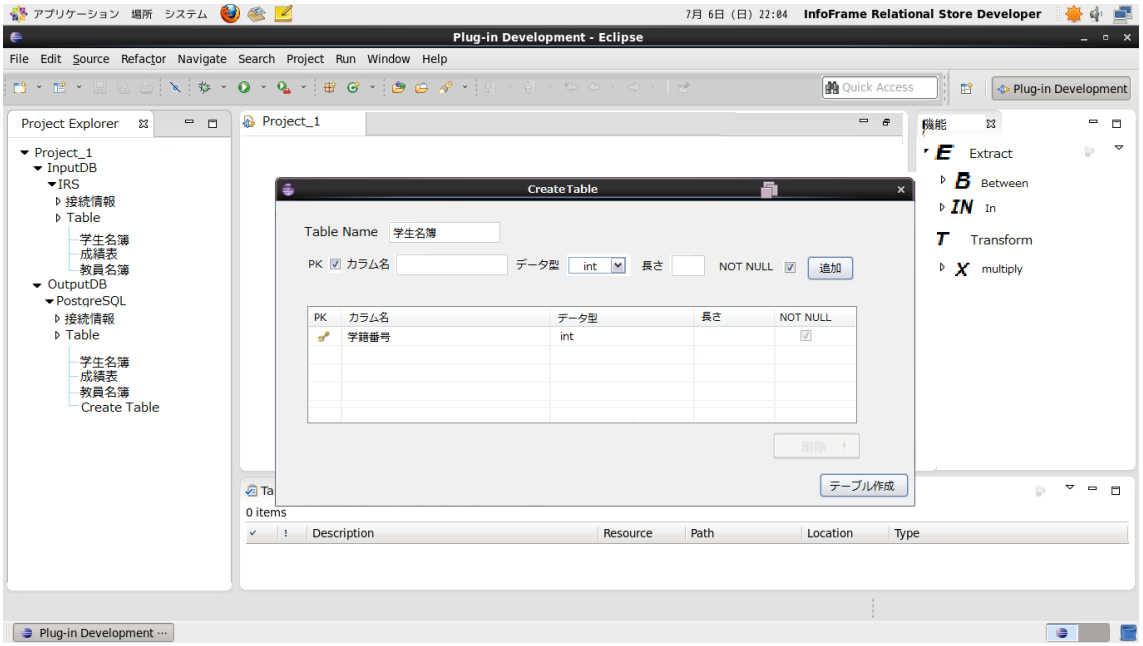


図 19 新規出力先テーブルの作成 3

また、追加したカラムはクリックで選択した後に、削除ボタンを使って削除することができる（図 16）。

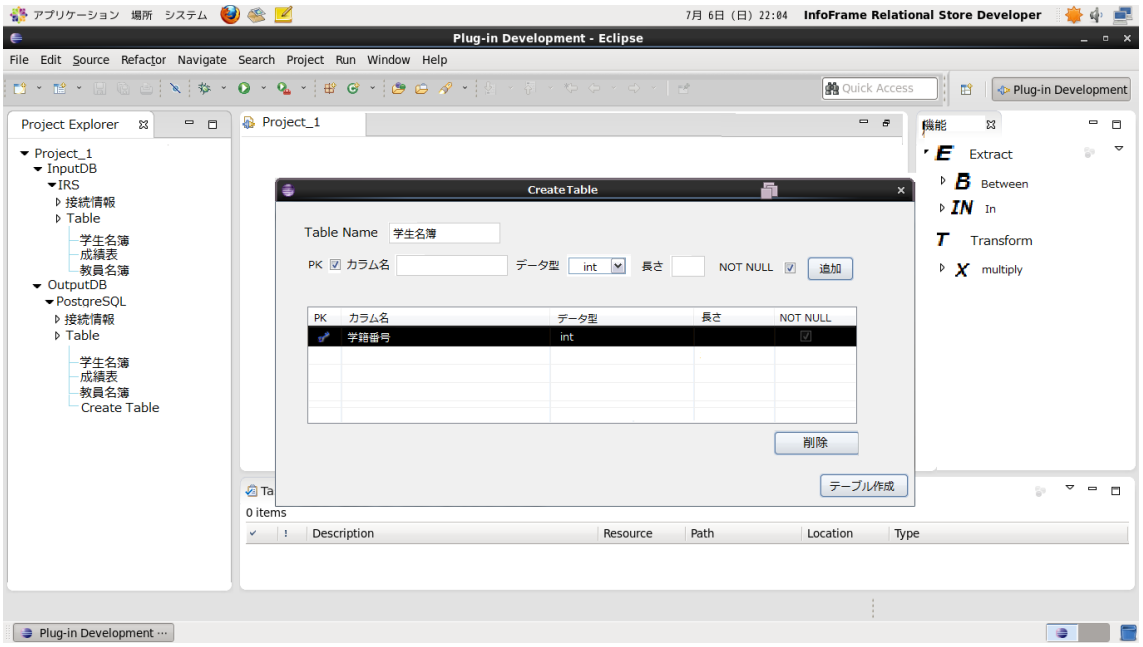


図 20 追加したカラムの選択、削除

次に、Input、output のテーブルをドラック&ドロップした後に、抽出、加工関数を作業ウィンドウにドラック＆ドラップする（図 17)。

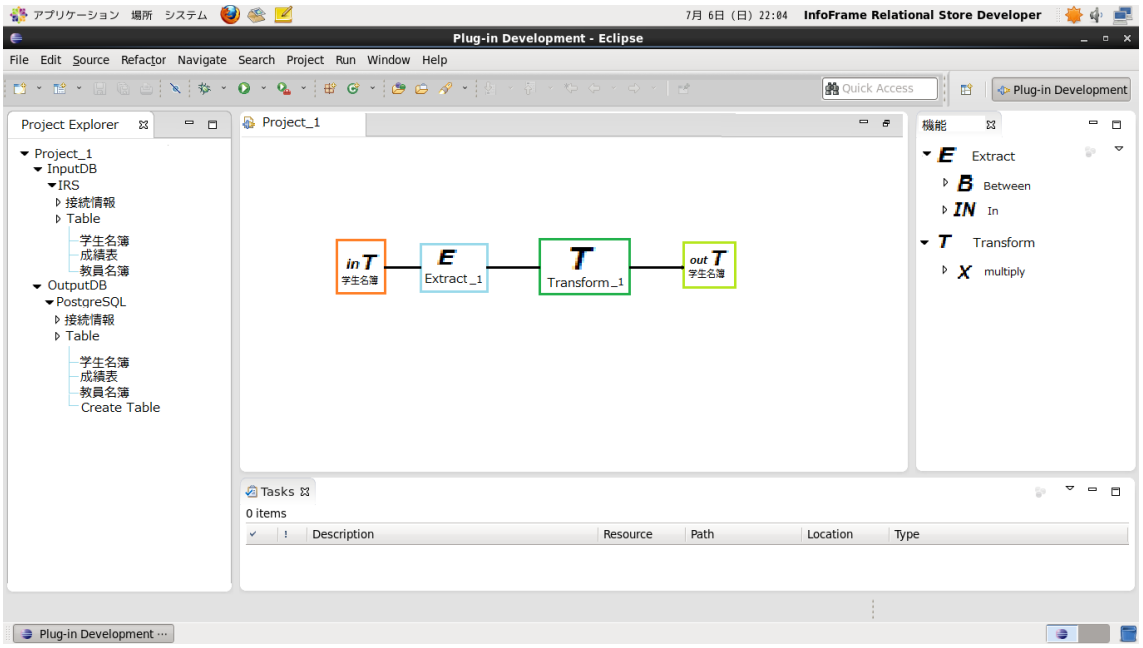


図 21 作業ウィンドウで紐づけする

また、複数の input テーブルに対応する際には図 18 のようにする予定である。

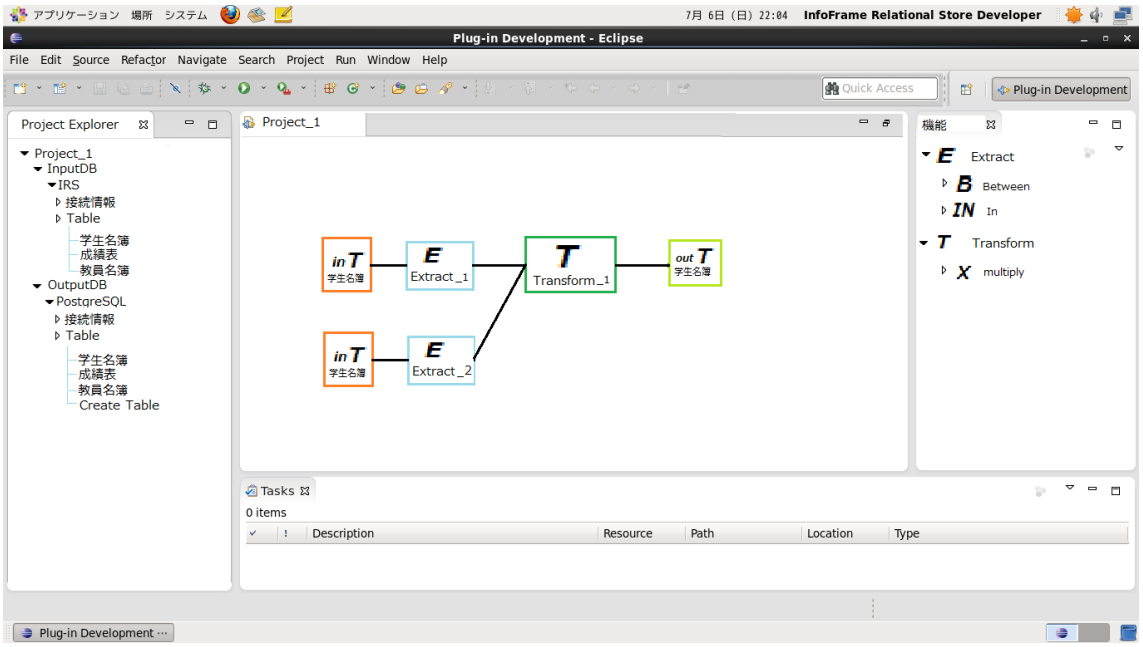


図 22 複数テーブル対応予定図

そして、抽出機能をダブルクリックすることで図 19 のような画面が表示され、inputDB からの抽出条件を設定する。

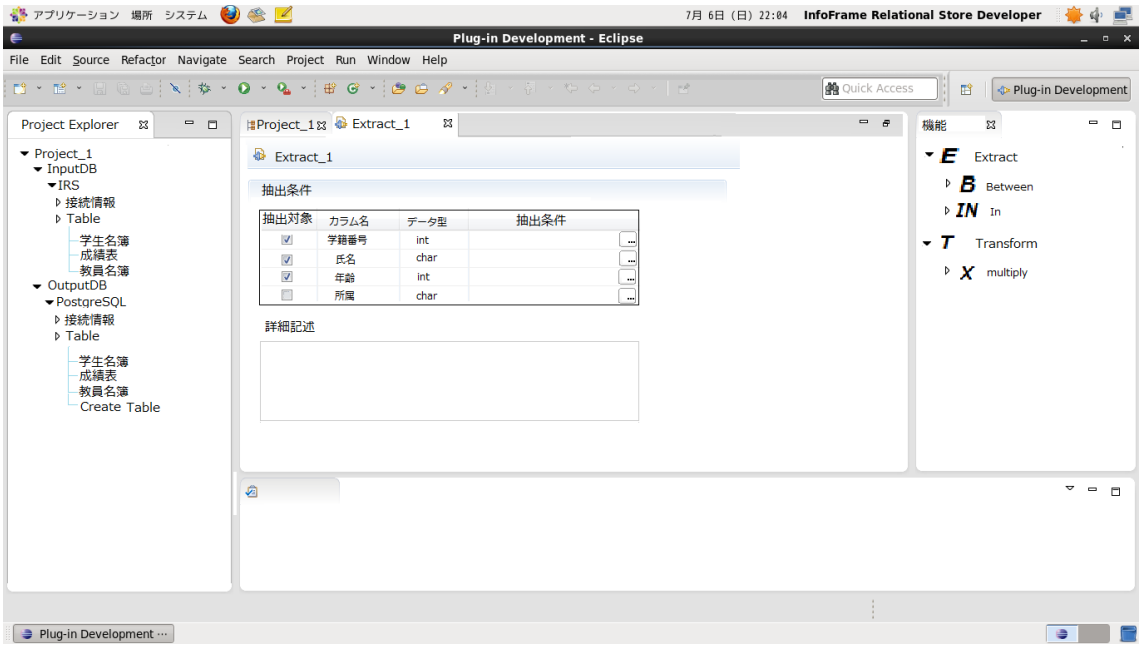


図 23 抽出機能ウィンドウ

抽出するカラムにチェックを入れ、抽出条件内にあるボタンをクリックすることでより条件を指定することができる(図 20)。

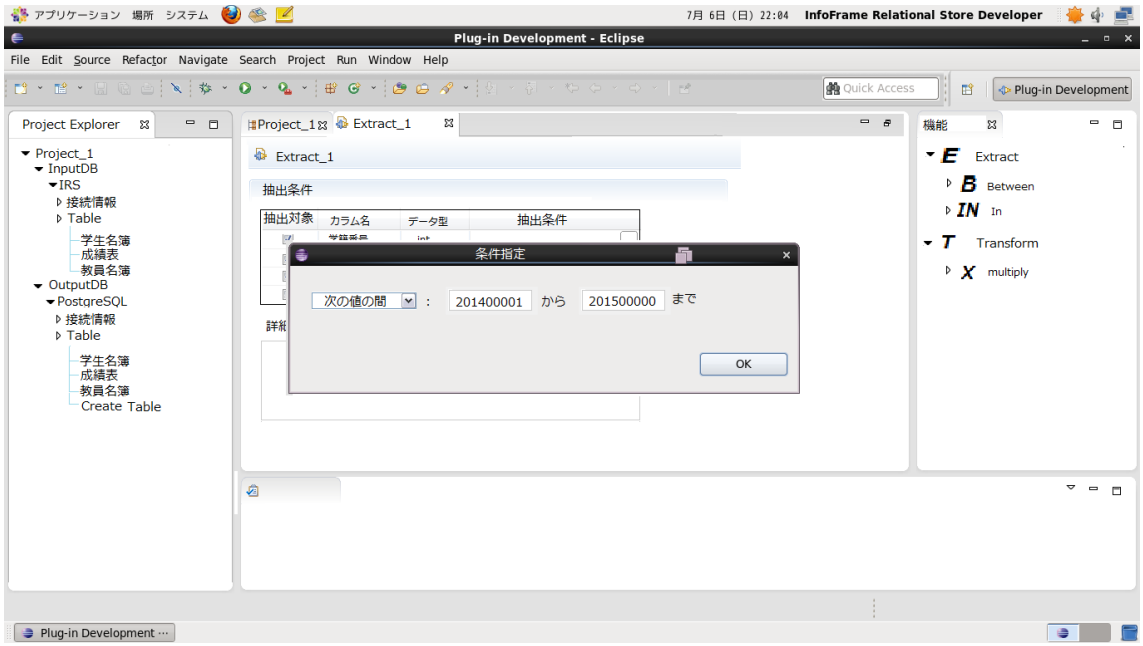


図 24 抽出の条件指定画面

指定後、「OK」ボタンを押すと、ポップアップウィンドウが閉じ、図 21 のように抽出条件が挿入されていることが確認できる。

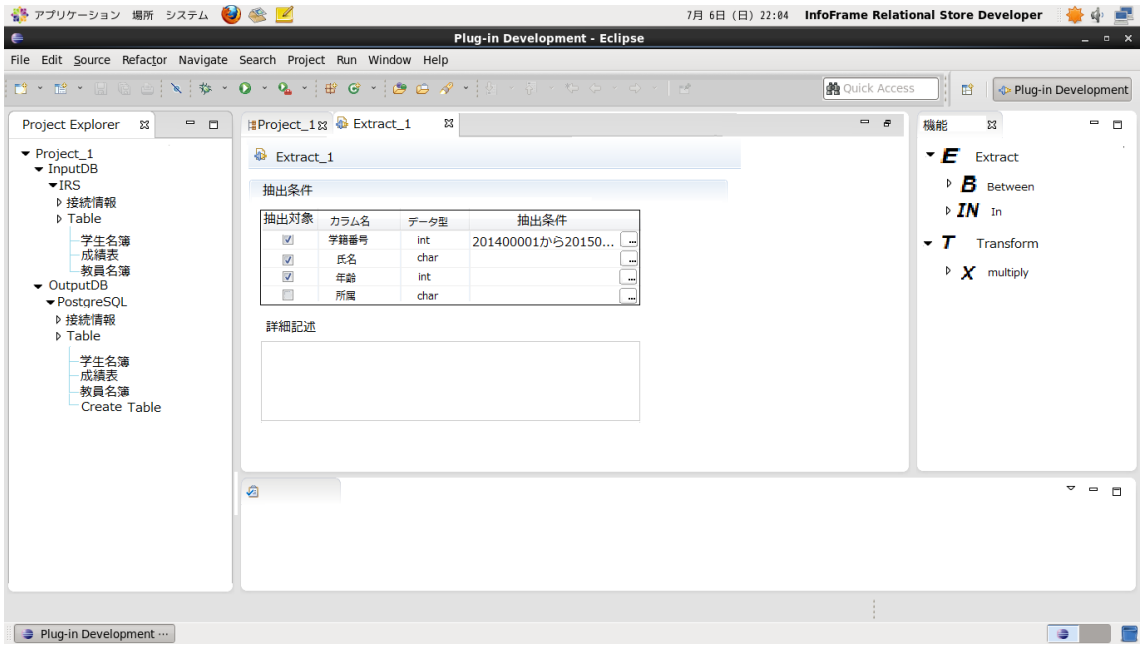


図 25 抽出機能の利用例

また、詳細記述の部分に抽出条件を打ち込むことで、より詳細に条件を指定することができる(図 22)。しかしながら、詳細記述で条件を指定された場合は図 19, 20 での条件指定が

無効になり、詳細記述での条件指定のみが有効になる。

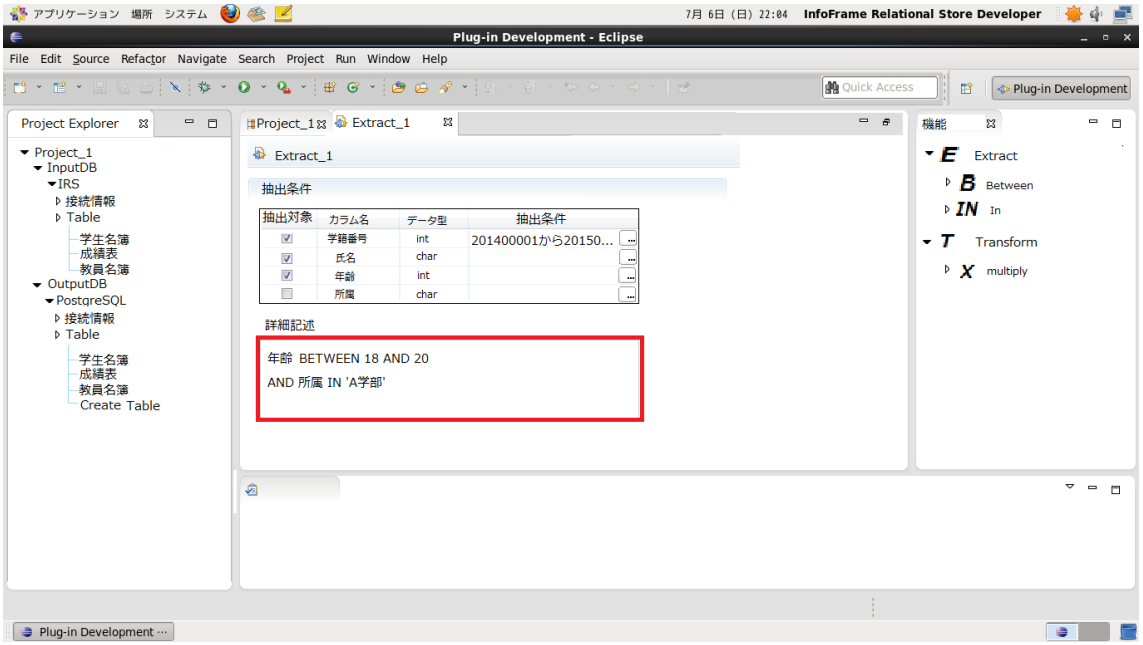


図 26　詳細記述利用例

加工関数も同様に作業ウィンドウでダブルクリックすることによって入出力間のカラムの対応付けを行うことができる(図 22)。加工関数の input は加工関数に紐づけられている、抽出関数の output になる。

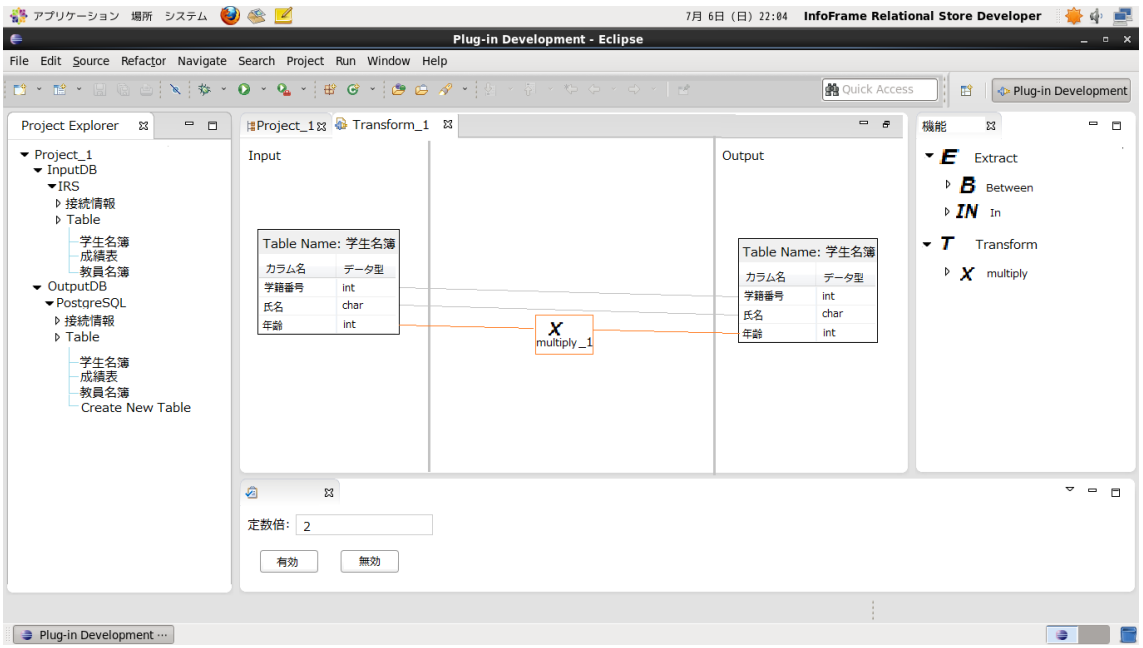


図 27 加工関数利用例

このウィンドウでも抽出関数と同様に加工機能を紐づけの間に置くことで詳細な加工機能を利用することができるようになる予定である。

このように DB に接続し、テーブルを取得、テーブルの抽出条件、テーブル間の関連付け、加工を行う。その後にプロジェクトを実行すると、出力先のテーブルに入力元のテーブルが移行され、成功した場合は図 24 のような画面が表示される。

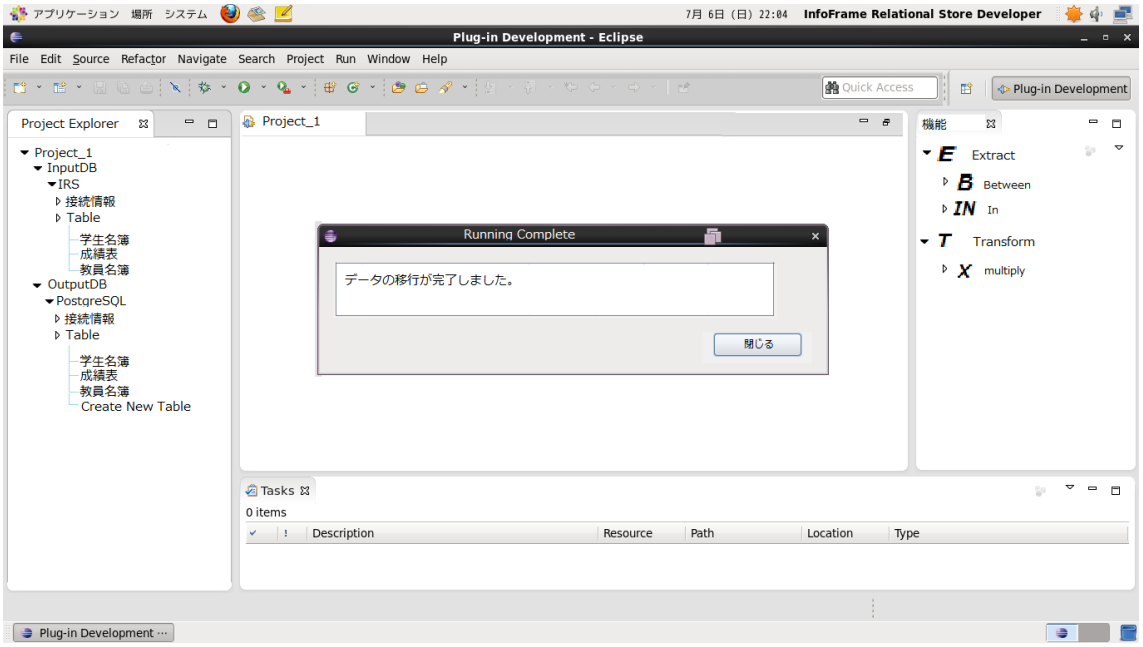


図 28 データ移行の完了

また失敗した場合は、失敗した原因に応じて図 25, 26 のような画面が表示される。

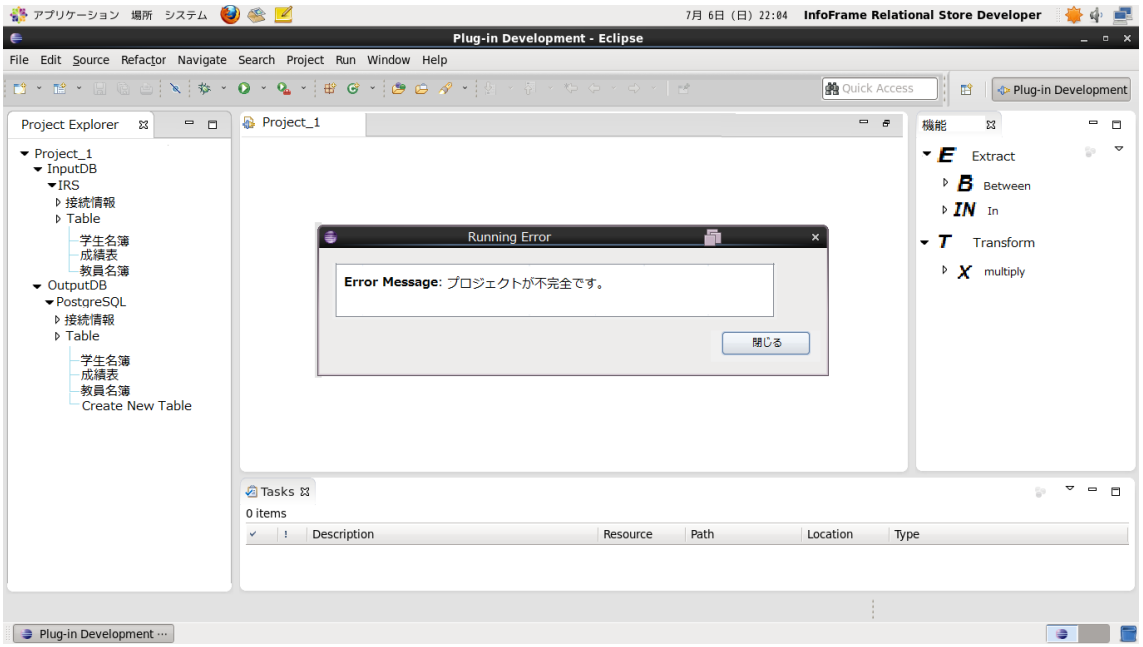


図 29 プロジェクトが不完全な場合のエラー

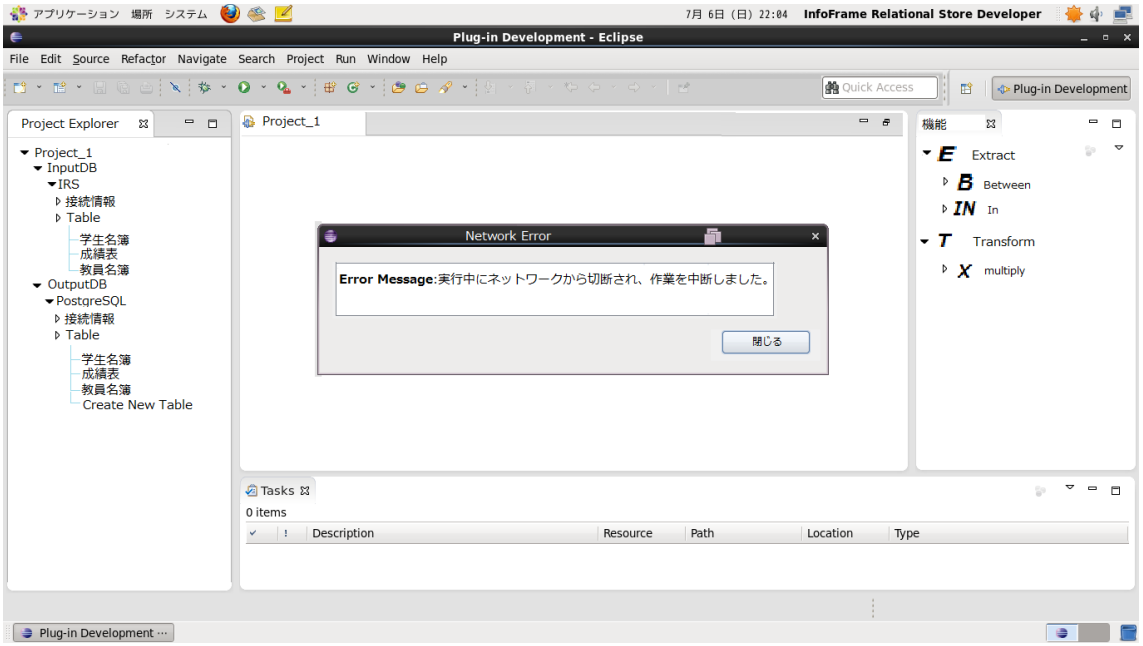


図 30　NetWork エラー

第５章 内部設計フェーズ

本章では内部設計フェーズで作成した成果物について述べる。成果物は、プログラム詳細仕様(Javadoc)である。

第１節　プログラム詳細仕様(Javadoc)

プログラム詳細仕様については Javadoc を作成したので、そちらを参考にされたし。Javadoc は納品フォルダ内の Javadoc フォルダに配置した。

第６章 コーディングフェーズ

本章ではコーディングフェーズで作成した成果物について述べる。成果物は、ソースコードである。

第１節　ソースコード

作成したソースコードは納品フォルダの
jp.ac.tsukuba.cs.itsoft.team_net.irs2db フォルダに作成したソースコードを Eclipse プロジェクトの形式で配置した。

第７章 テストフェーズ

本章ではテストフェーズで作成した成果物について述べる。成果物は、単体テスト報告書、機能テスト報告書、総合テスト報告書、非機能要件調査報告書である

第１節　　単体テスト報告書

本節では実施した単体テストについて報告する

目的：バグを検出し、ソフトウェアの品質を保つため

方法：JUnit を使用し、public のメソッドに対してテストケースを作成し、単体テストを実施した。

単体テストで使用したテストケースは納品フォルダの単体テストケースフォルダ内に配置した。ファイル名が「単体テストテストケース一覧 S1」となっているものは第一イテレーションで行ったテストである。ファイル名の後半にテスト対象の機能の名前が付いている。

「単体テストテストケース一覧 S2」となっているものは、第二イテレーションで行ったテストである。ファイル名の後半にテスト対象である第二イテレーションの要件定義の機能番号が付いている。第二イテレーションでは第一イテレーションで行った単体テストも行った。

第２節　　第１イテレーション機能テスト報告書

本説では実施した機能テストについて報告する

目的：「第１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、「第１イテレーション機能詳細一覧」を元にテストを作成し、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

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	左に同じ	合	2014/10/13

		る。			
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	左に同じ	合	2014/10/13

		表示されます			
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	作業ウィンド	左に同じ	合	2014/10/11

	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。	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。			
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
7-8	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	左に同じ	合	2014/09/29

		ンに使用したログイン情報が表示される。			

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できているか確かめる。	正常にテーブルが取得でき、テーブルリストに表示されている	左に同じ	合	2014/09/25
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、	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	否	2014/09/28 テーブル取得はデータベー

	再度テーブル情報を取得する		れる		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か確かめる。	新しくテーブルが作成され、テーブルリストが更新さ、作成されたテーブル情報がポップアップで表示される	左に同じ	合	2014/10/08
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	接続情報を入力するまでは、右クリ	左に同じ	合	2014/10/11

	作成を行おうとする	ックメニューにテーブル作成を表示させない			
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スが押せなくなる	否	2014/10/08 対処：現在の仕様の方がユーザにとって親切だと判断し、修正せず。
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」	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。			

	と線をつなぐ。				

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トが行われている。	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)
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 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	線が引けない	左に同じ	合	2014/10/01
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	Connect が選択され、	左に同じ	合	2014/10/11

	に既にアイコンがある状態で、「Connect」を選択する。	マウスカーソルが変化する。			
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。

第３節　　第２イテレーション機能テスト報告書

本説では実施した機能テストについて報告する

目的：「第 2 イテレーション要件定義」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、「第 2 イテレーション要件定義」を元にテストを作成し、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	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 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
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	左に同じ	合	2014/11/27

		たテーブルが作成される			
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 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
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	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 の varchar 型のカラムを持ったテーブルが作成される			
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 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
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	左に同じ	合	2014/11/27

		ラムを持ったテーブルが作成される			
	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データをインサートする	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	2014/11/28
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、終了を告知するポップアップが表示されることを確かめる。	正常に ETL 作業が終了した時点で終了を告知するポップアップが表示される	左に同じ	合	2014/11/27

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	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	左に同じ	合	2014/11/21

	報を取得し、取得できているか確かめる	る			
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アップが表示されるか確かめる。	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
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	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	左に同じ	合	2014/11/21

	表示される か確かめる	示される			

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 は空欄で表示される。	左に同じ。	合	2014/12/02

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	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

	超える場合、線が引けないことを確かめる				
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か確かめる	正しい実行結果が出る	左に同じ	合	2014/12/01

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	「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

	行う				
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	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	左に同じ	合	2014/11/27

	確かめる。	ップが表示される。			
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

第４節　　第１イテレーション総合テスト報告書

本節では、実施した総合テストについて報告する。

第１項　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る。

第２項　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表3に示す。

表 2　テスト環境

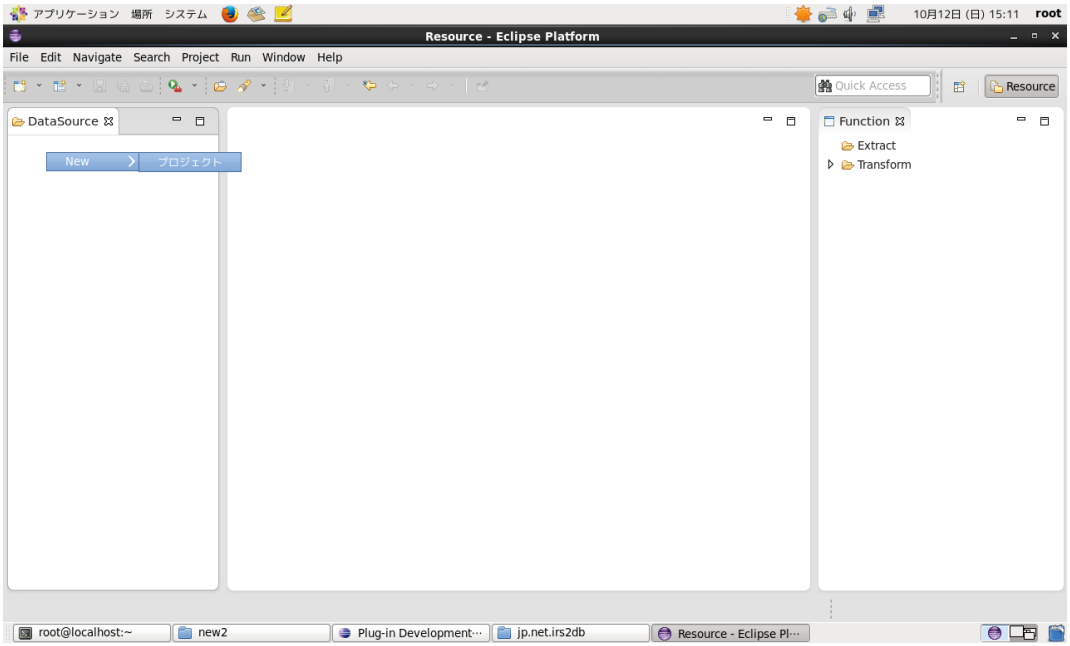
ハードウェア環境	
メモリ	2 GB
ハードディスク	40 GB
ネットワーク	ホストの IP アドレスを共有して使用
ソフトウェア環境	

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

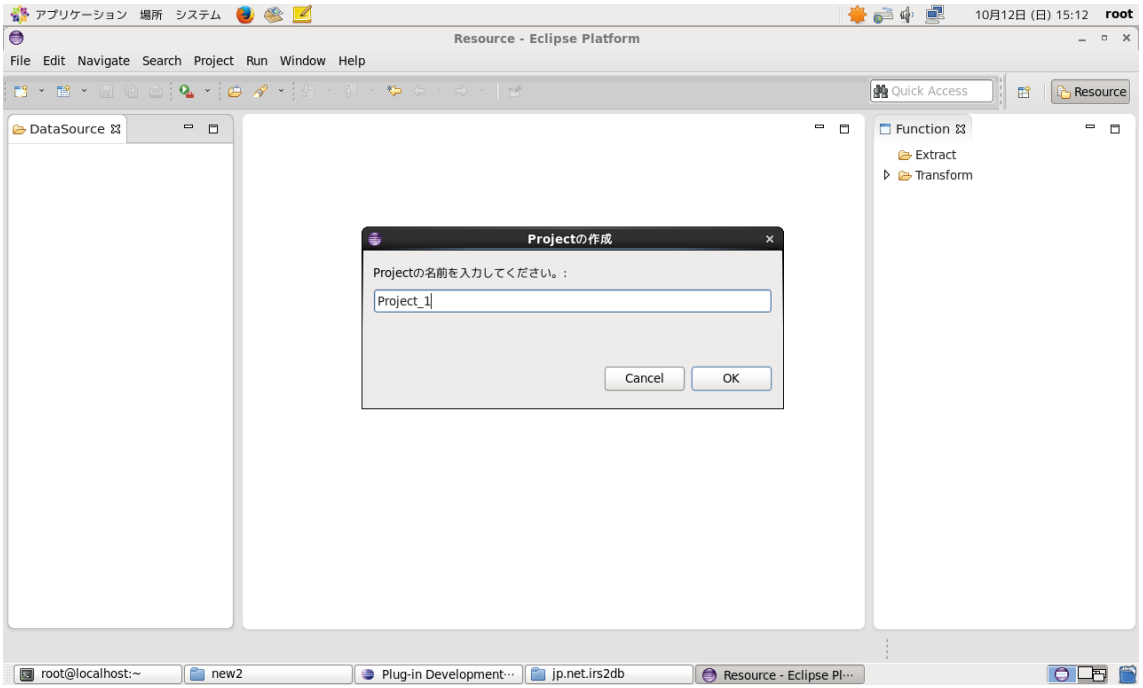
第３項　テスト操作と結果

1. **プロジェクト作成**

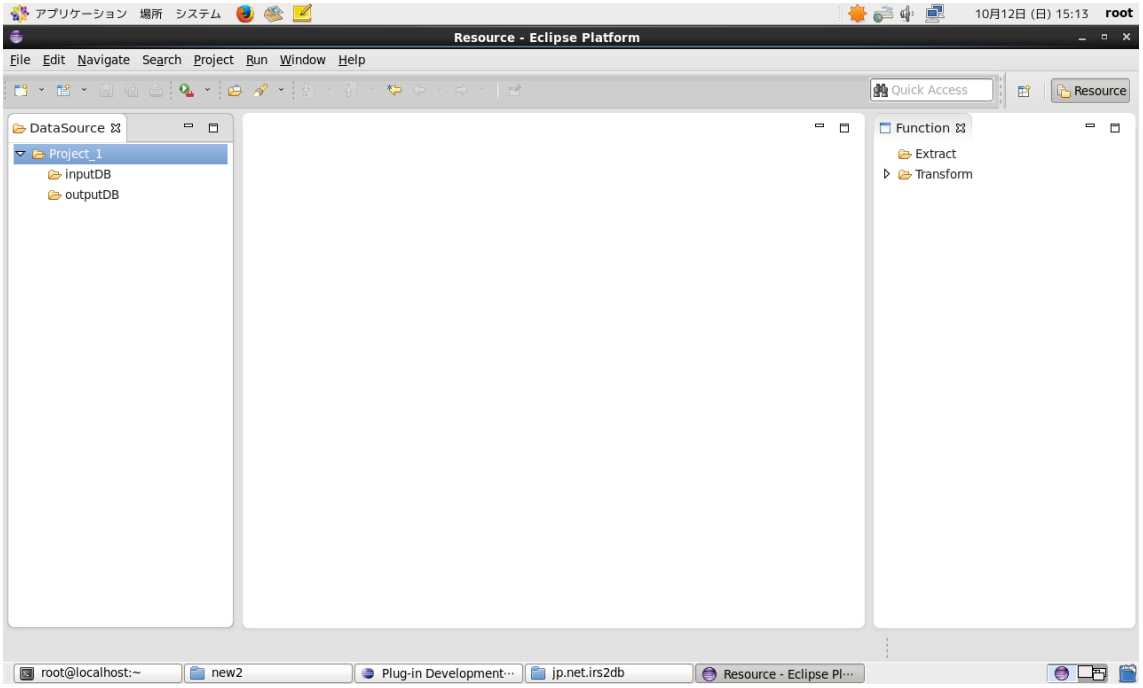
操作１:DataSource ビューで右クリックして、プロジェクトを選択する。



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。

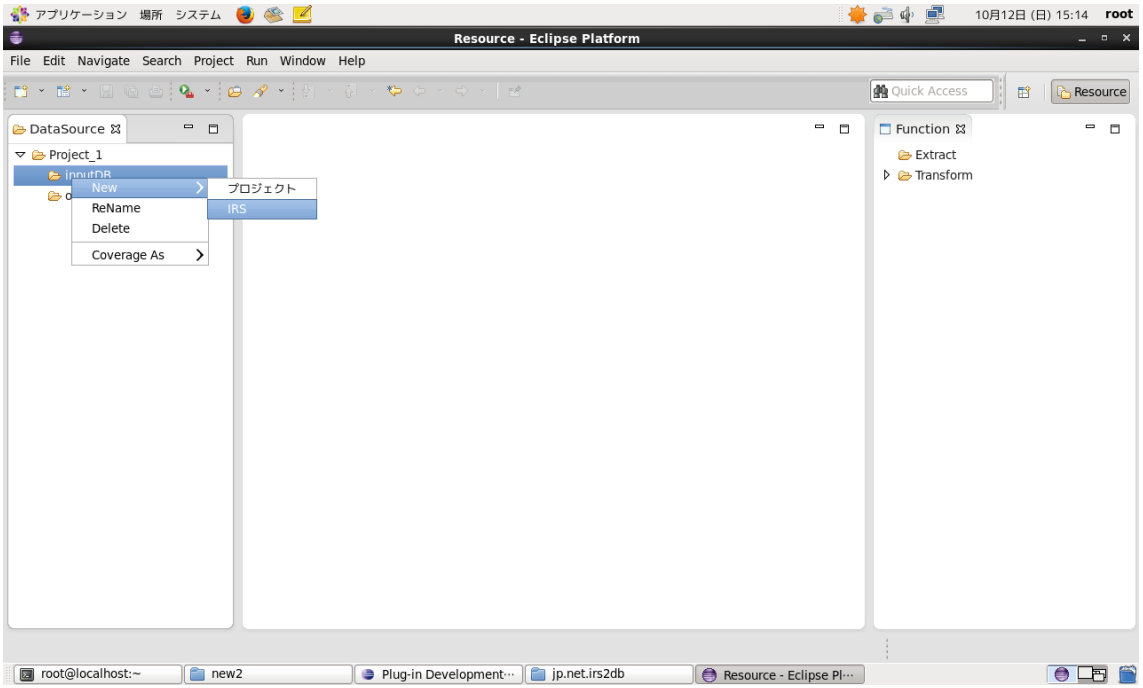


結果：「Project_1」というプロジェクトが新規作成され、「inputDB」と「outputDB」が表示される。

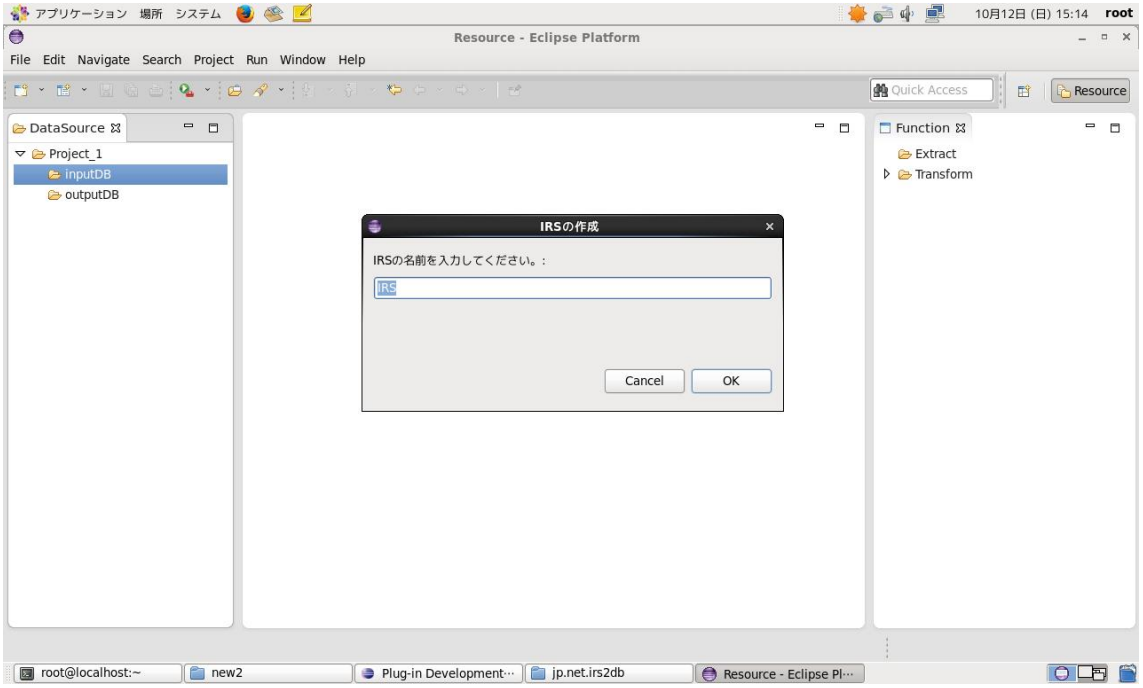


1.1. IRS 接続

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



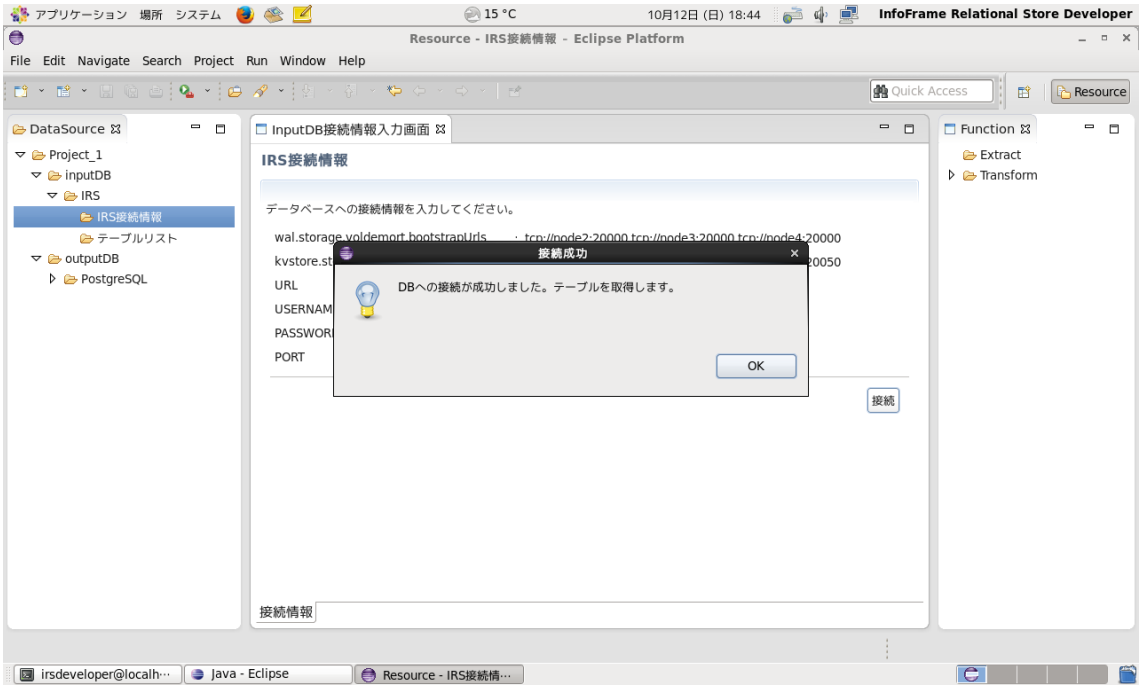
操作 2：「IRS 作成画面」で IRS の名前を入力して、「OK」ボタンを押す。



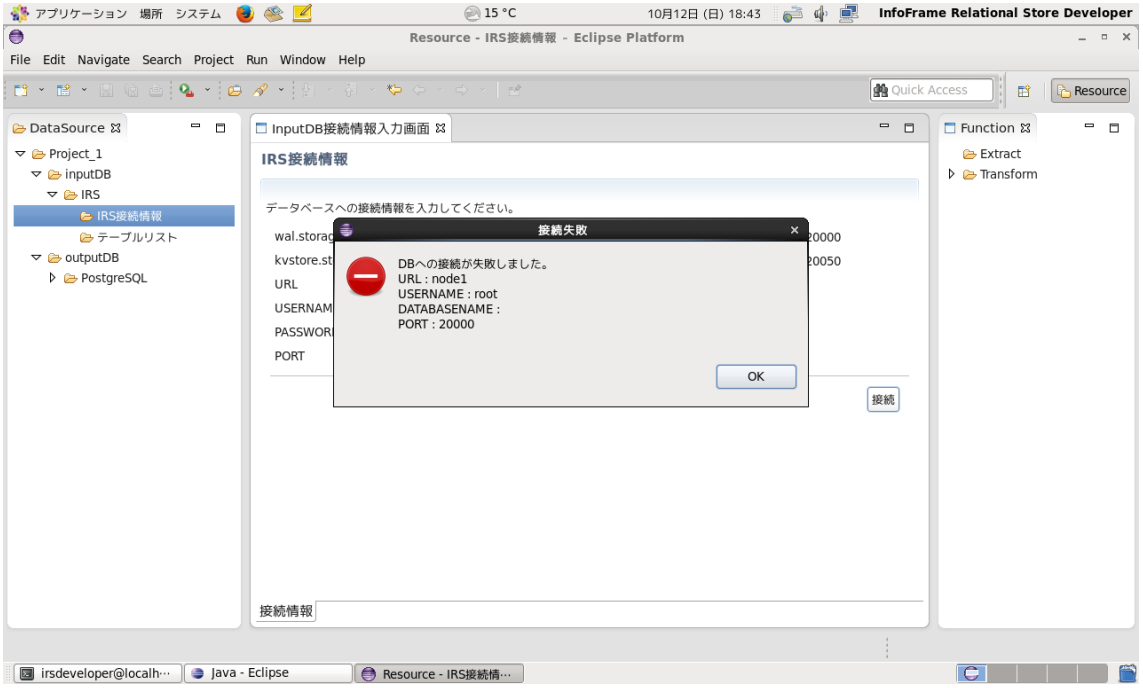
操作 3：作成した「IRS」を選択し、「IRS 接続情報」をダブルクリックする。

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る。

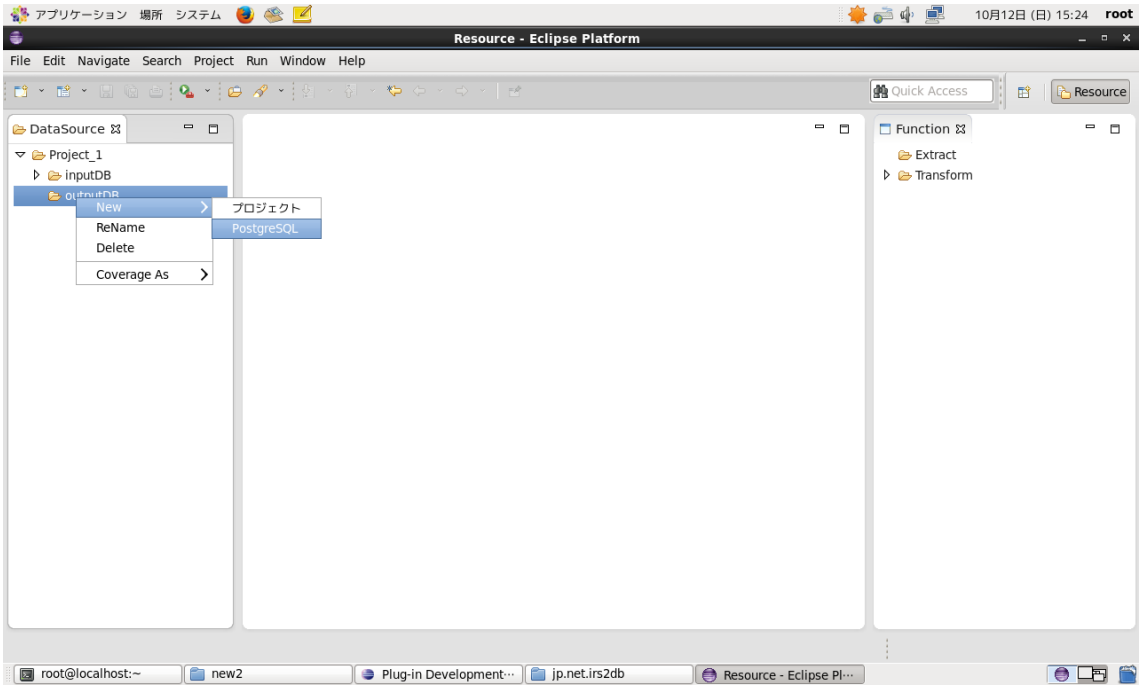


結果 B：接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示される。

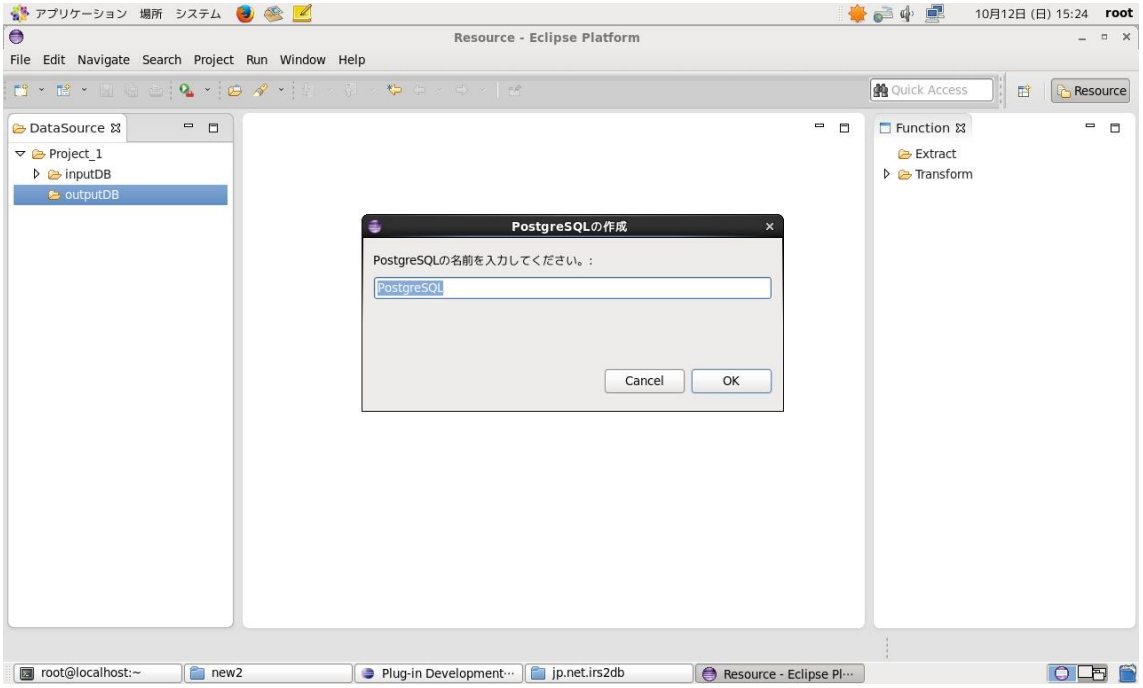


1.2. PostgreSQL 接続

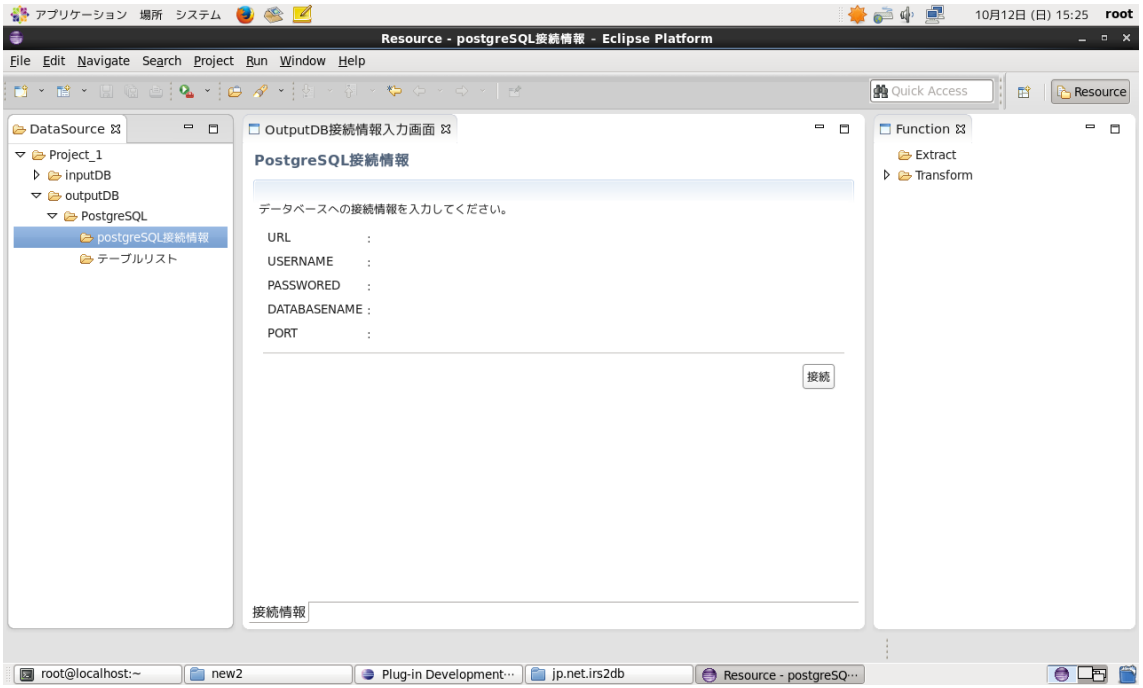
操作 1：「outputDB」を右クリックして、「New」→「PostgreSQL」を選択する。



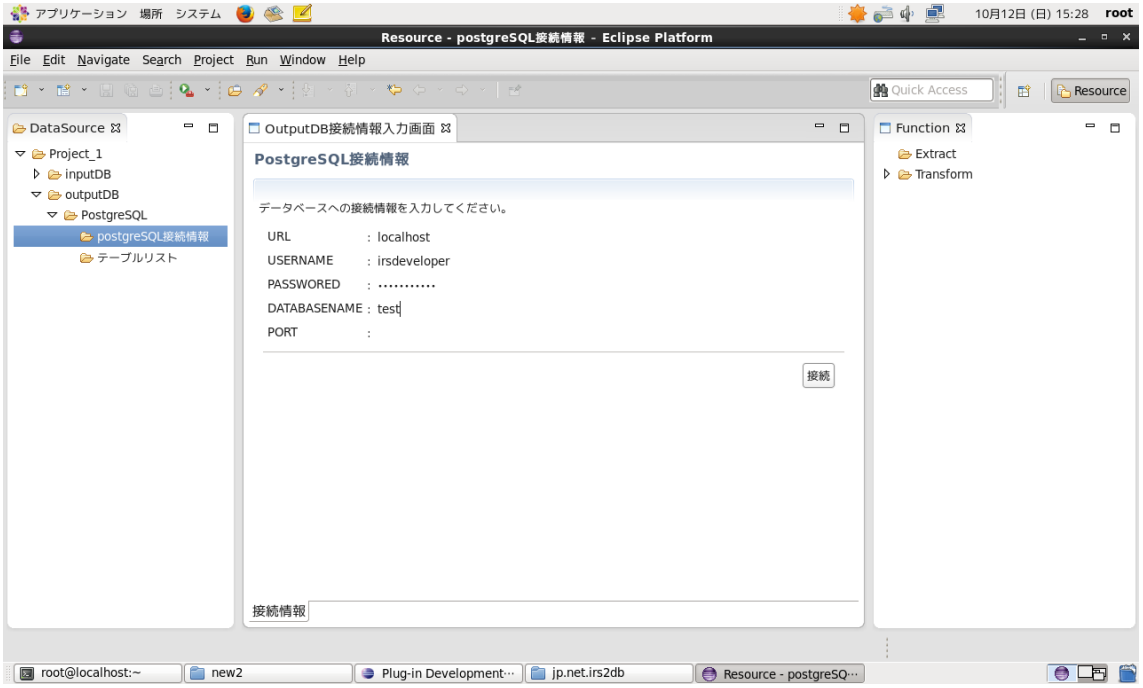
操作 2：「PostgreSQL 作成画面」で PostgreSQL の名前を入力して、「OK」ボタンを押す。



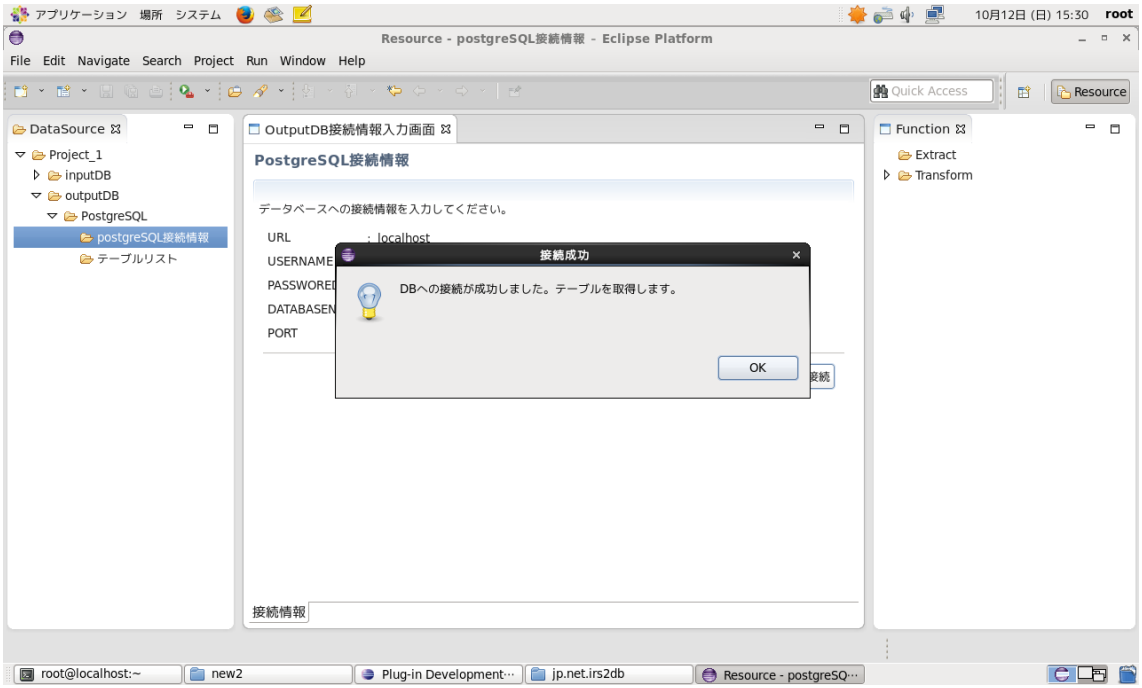
操作 3：作成した「PostgreSQL」を選択し、「postgreSQL 接続情報」をダブルクリックする。



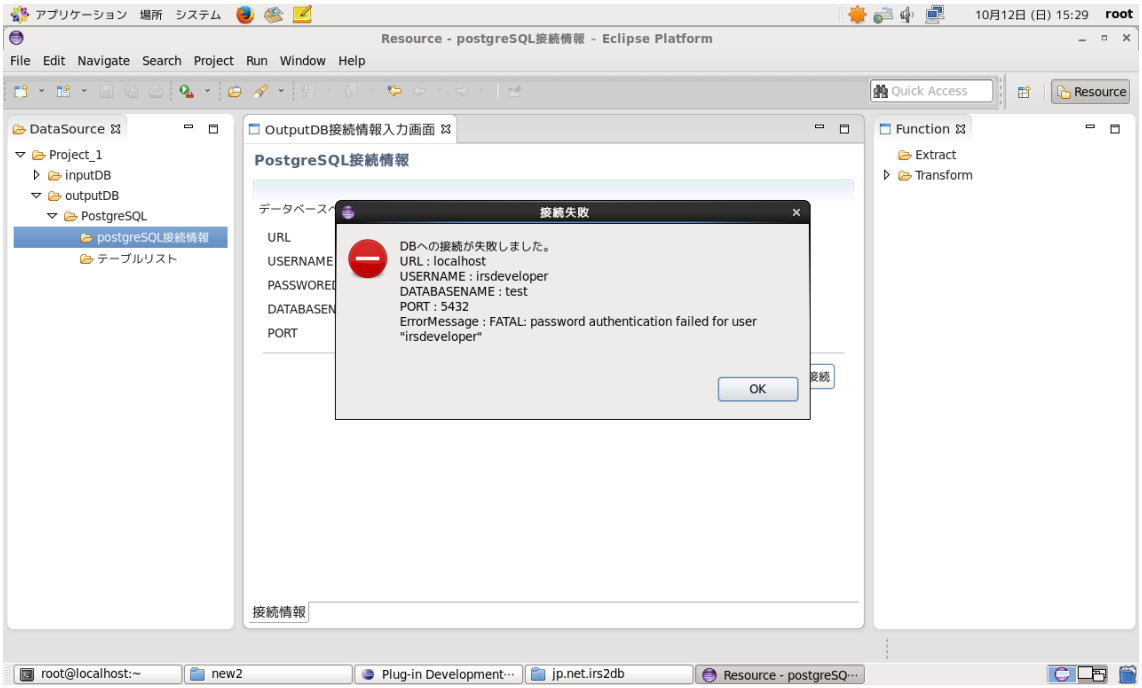
操作 4：「OutputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押す。



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る。



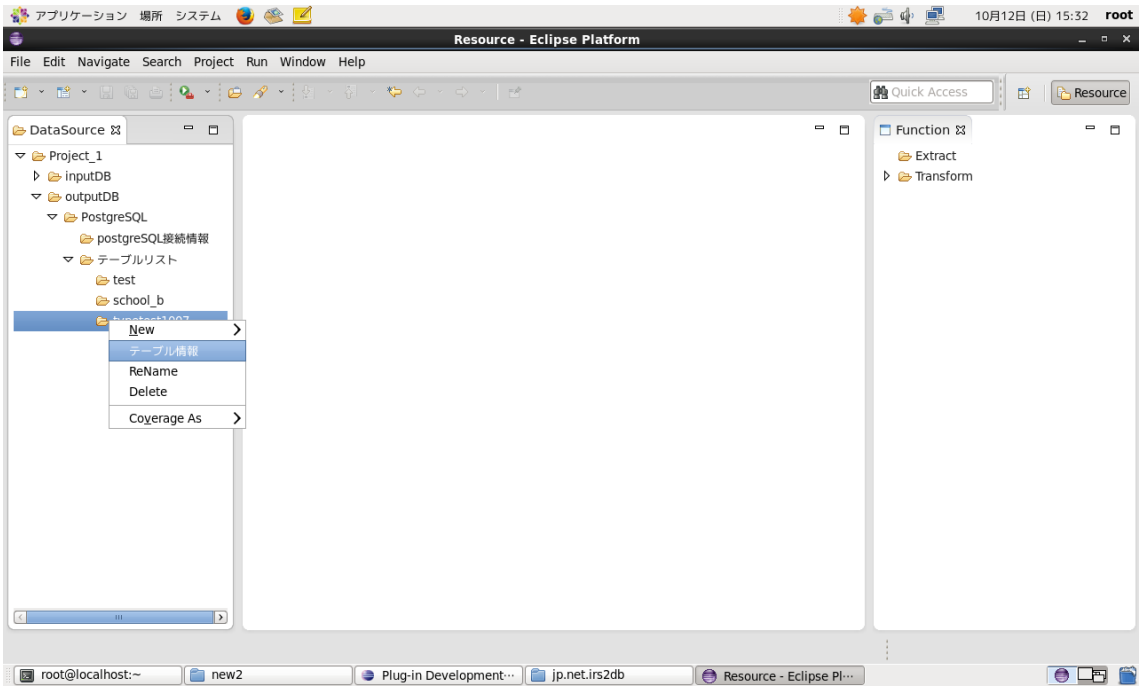
結果 B：接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示される。



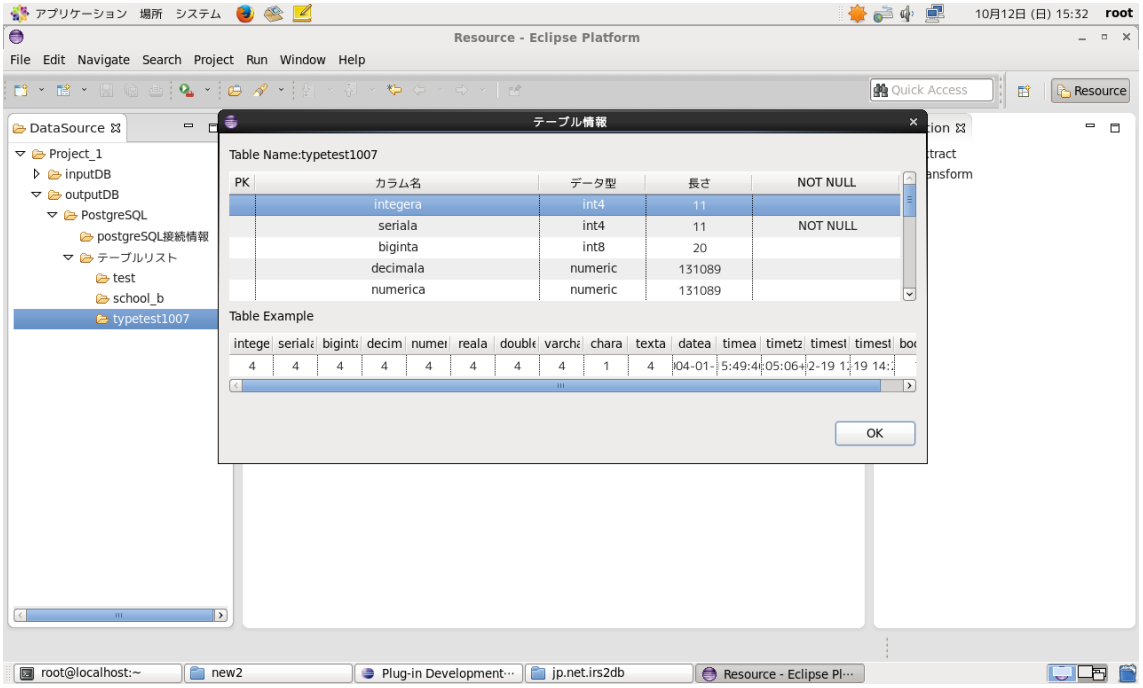
1.3. テーブル情報の表示

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」を選択する。

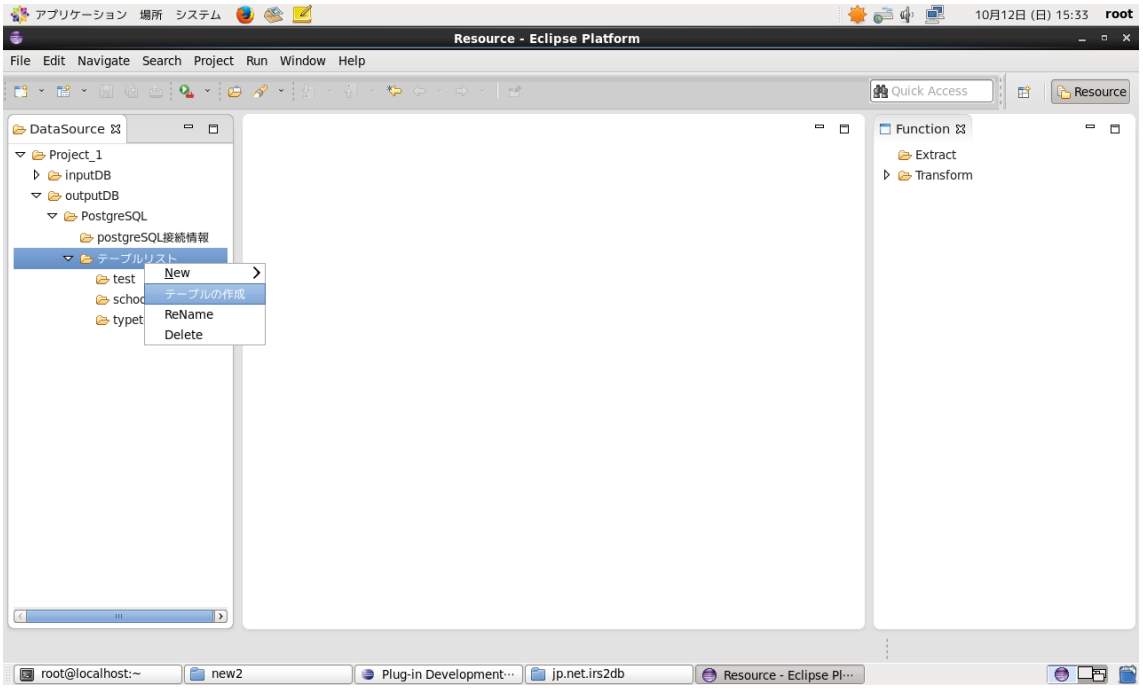


結果：テーブル情報が表示される。

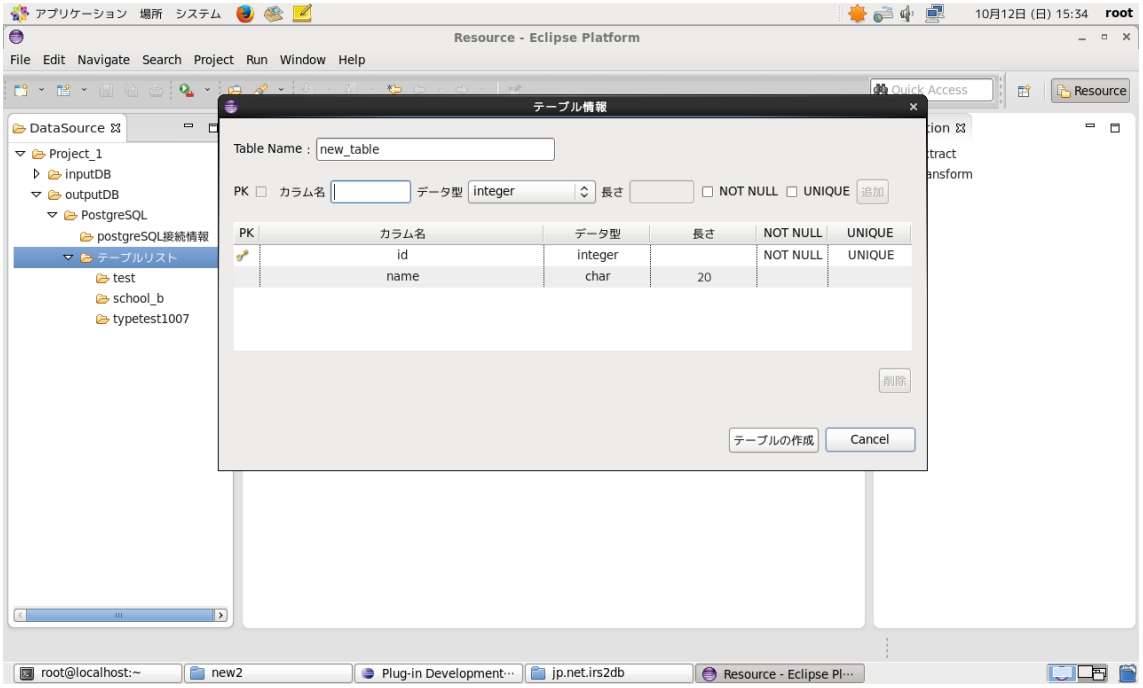


1.4. PostgreSQL に新規テーブルを作成

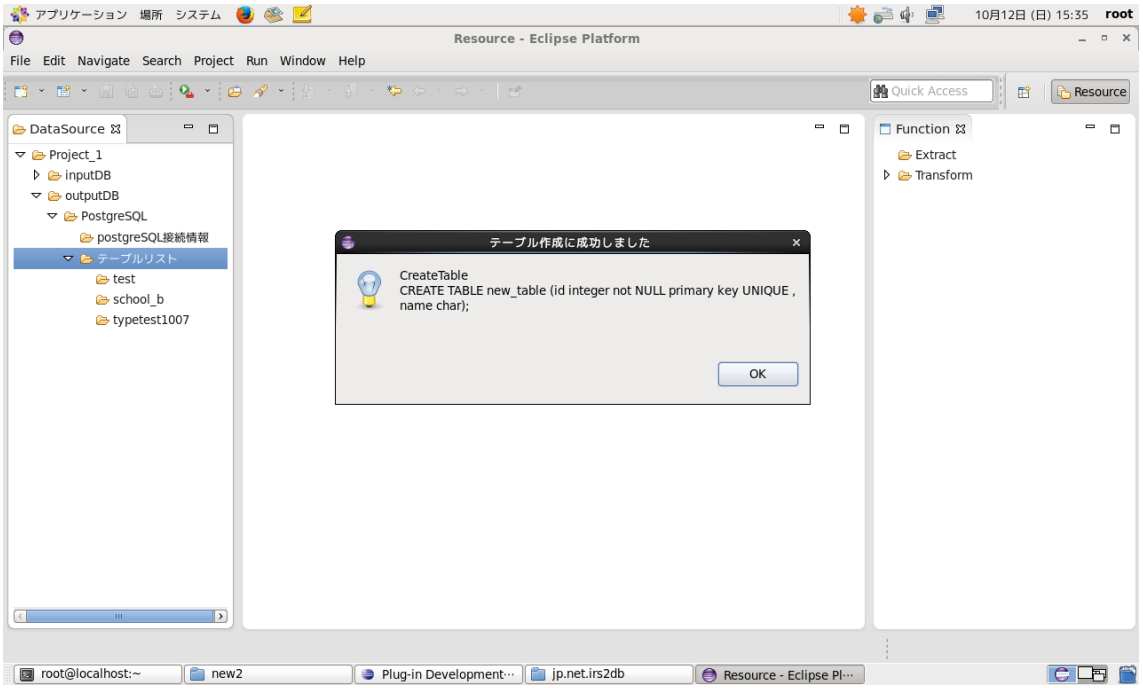
操作１:「テーブルリスト」を右クリックして、「テーブルの作成」を選択する。

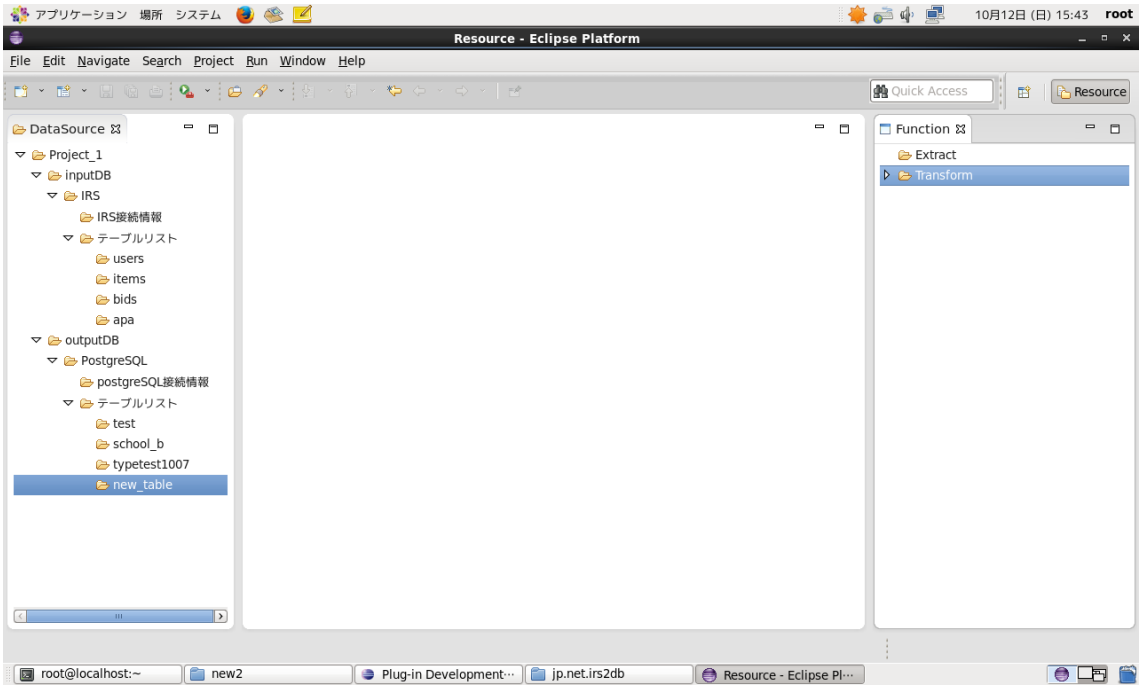


操作２:「テーブル情報」で作成するテーブルの情報を入力して、「テーブルの作成」ボタンを押す。

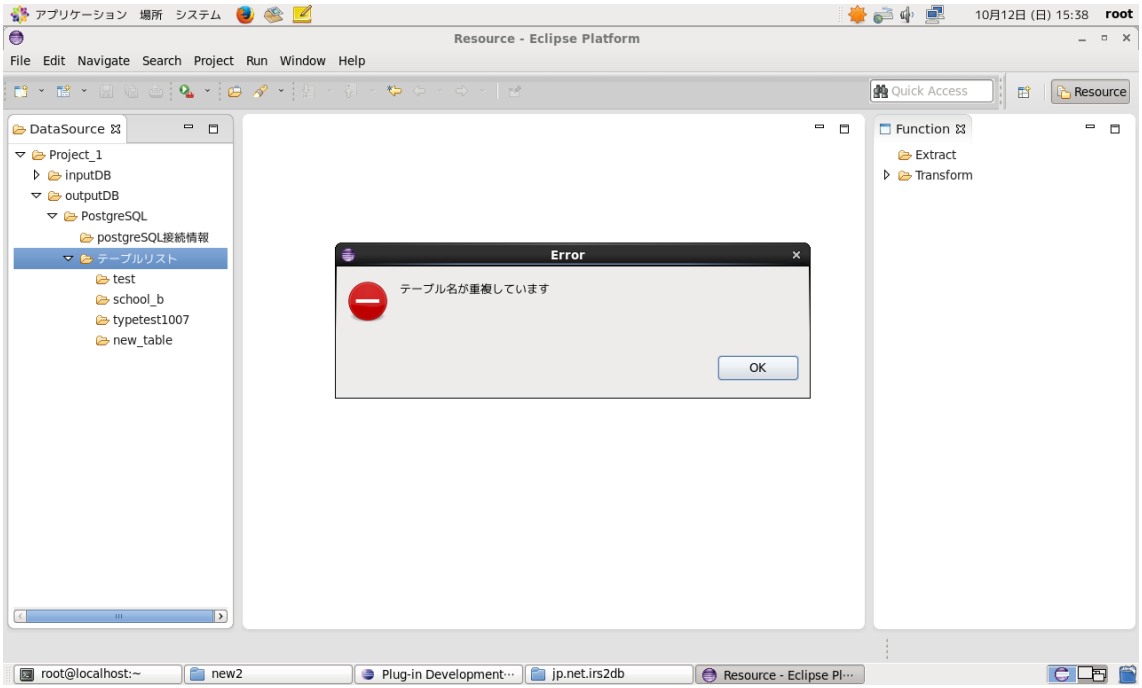


結果 A：正しく作成できた場合、「テーブルの作成に成功しました」ポップアップが表示されている。再接続すると、テーブルリストにテーブル名が追加される。



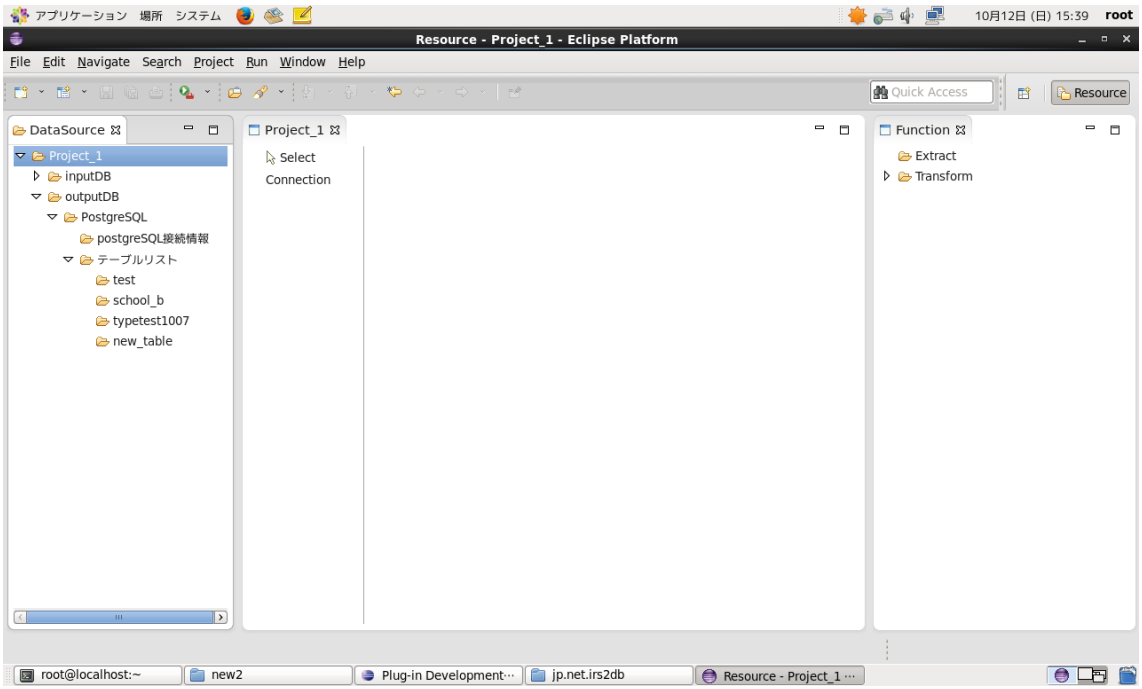


結果 B：デーブル作成が失敗した場合は、エラー画面が表示される。



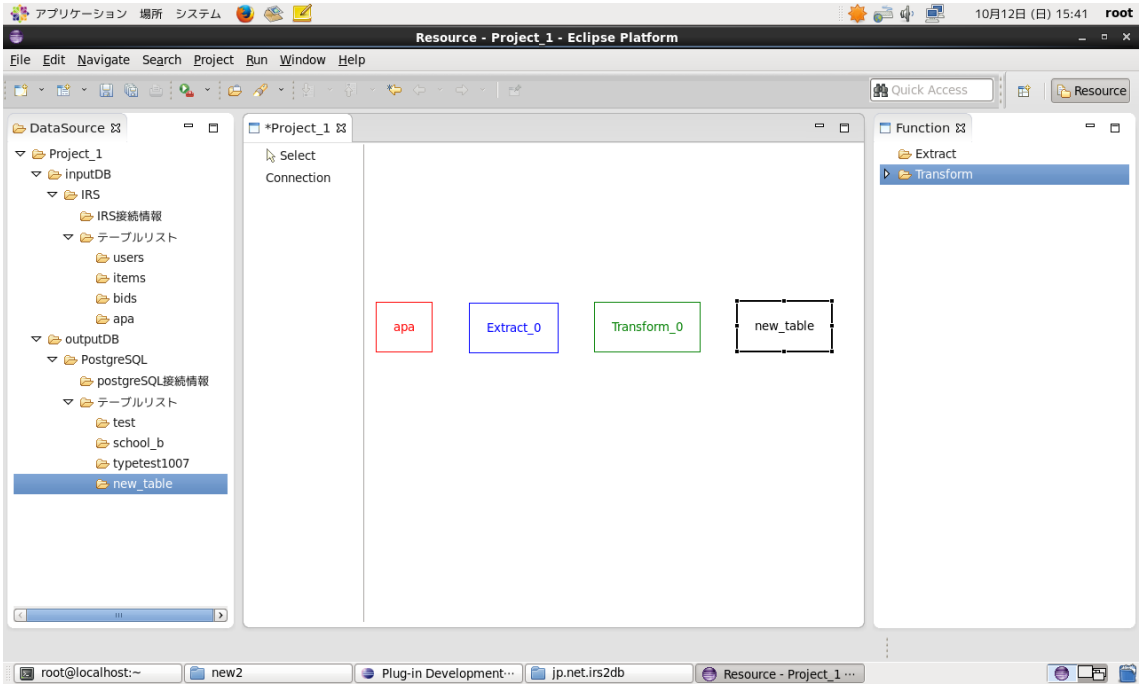
1.5. ETL 作業の編集

操作 1 : プロジェクト名をダブルクリックすることで、作業ウィンドウが開く。



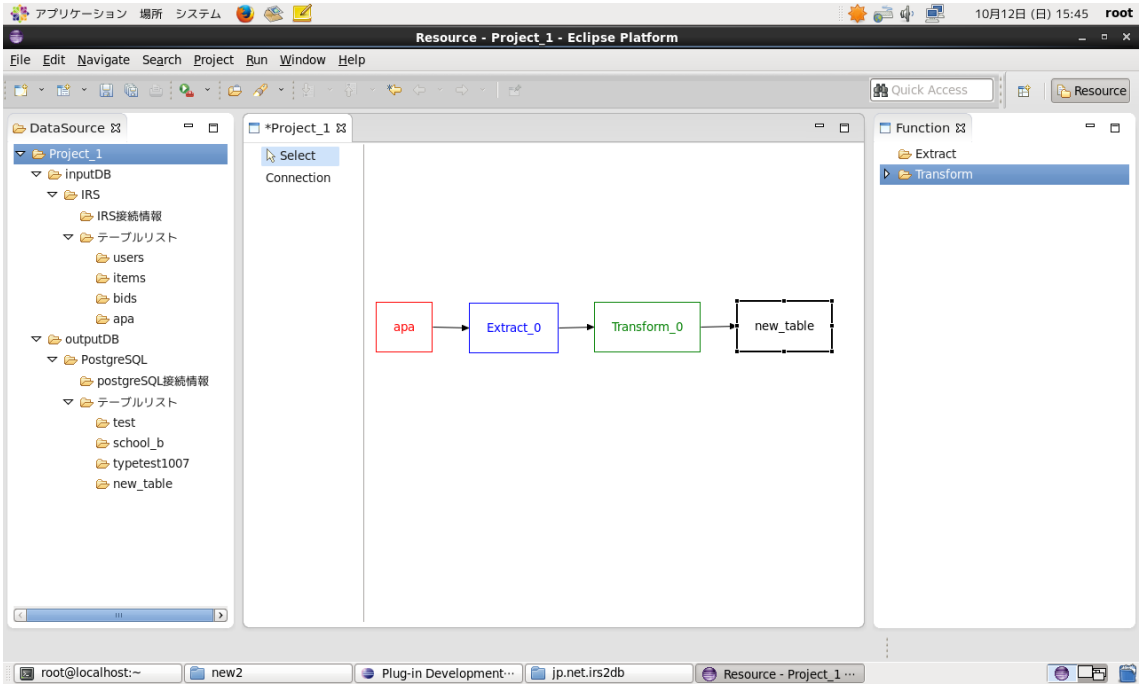
操作 2 : 使用するテーブルやファンクションなどを作業ウィンドウにドラッグする。

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る。

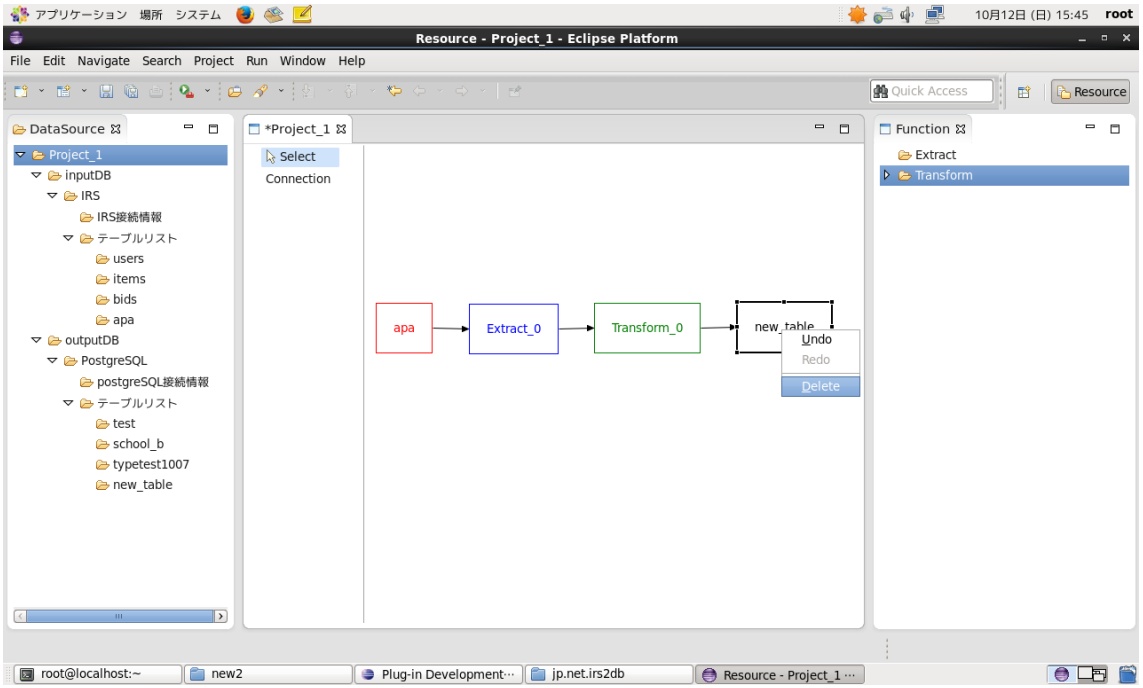


操作３：左側の「Connection」を選択して、作業ウィンドウ内のアイコンを紐付けする。

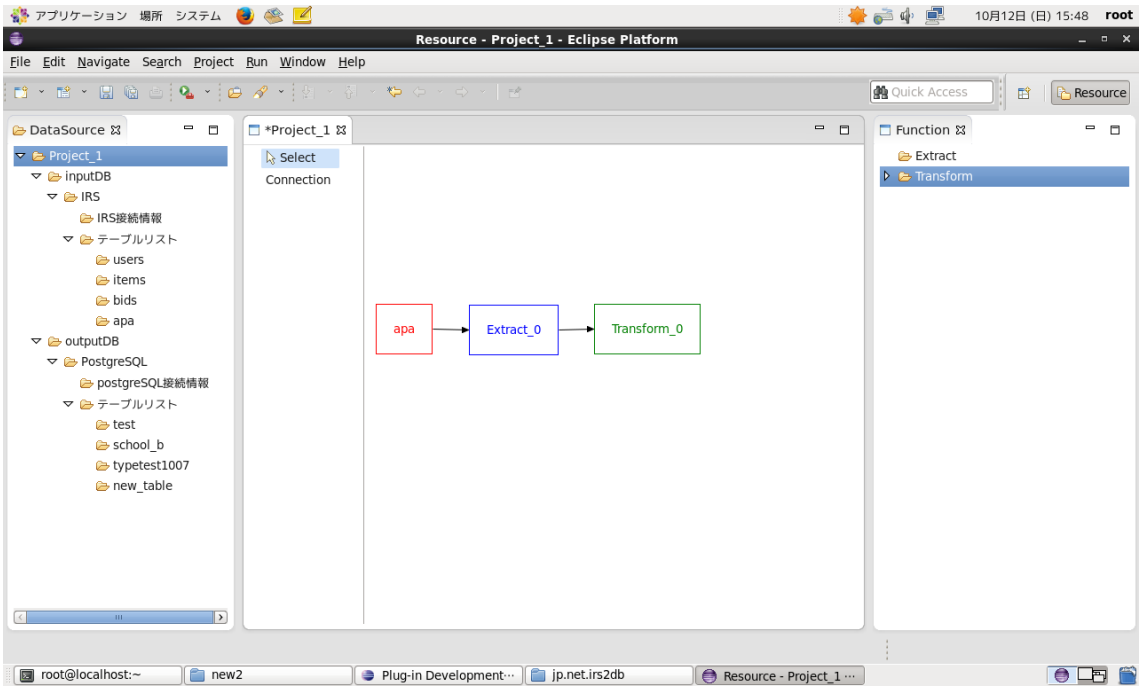
結果：テーブルと各ファンクションの対応関係が指定されている。



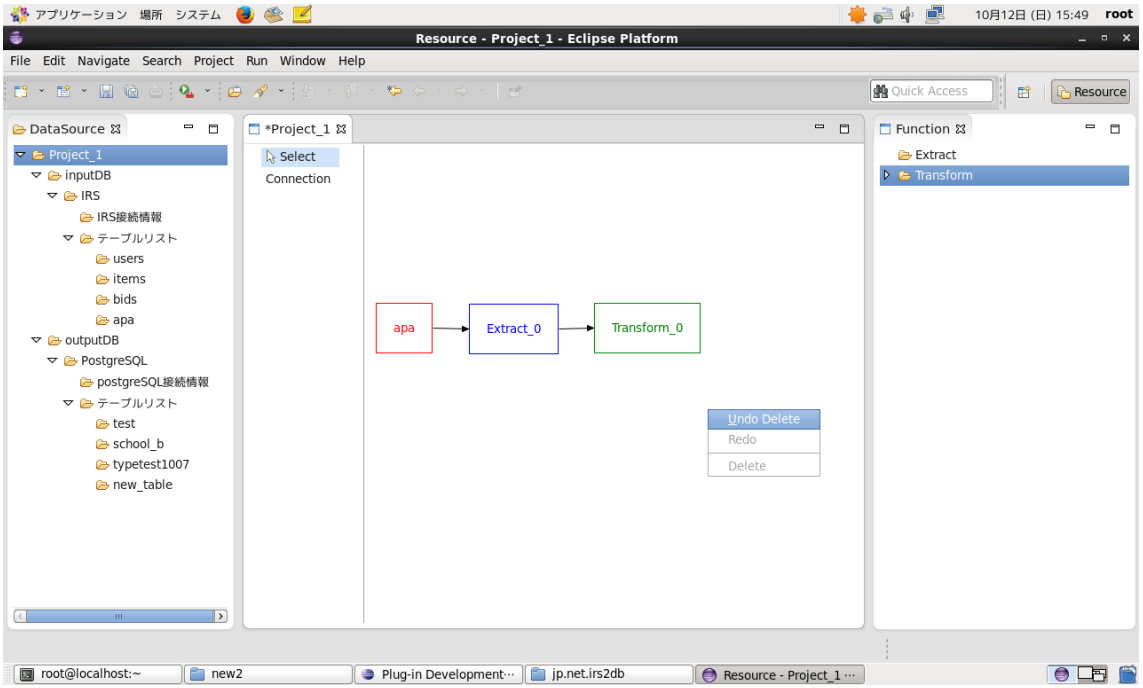
操作４：アイコンを右クリックして、「Delete」を選択する。



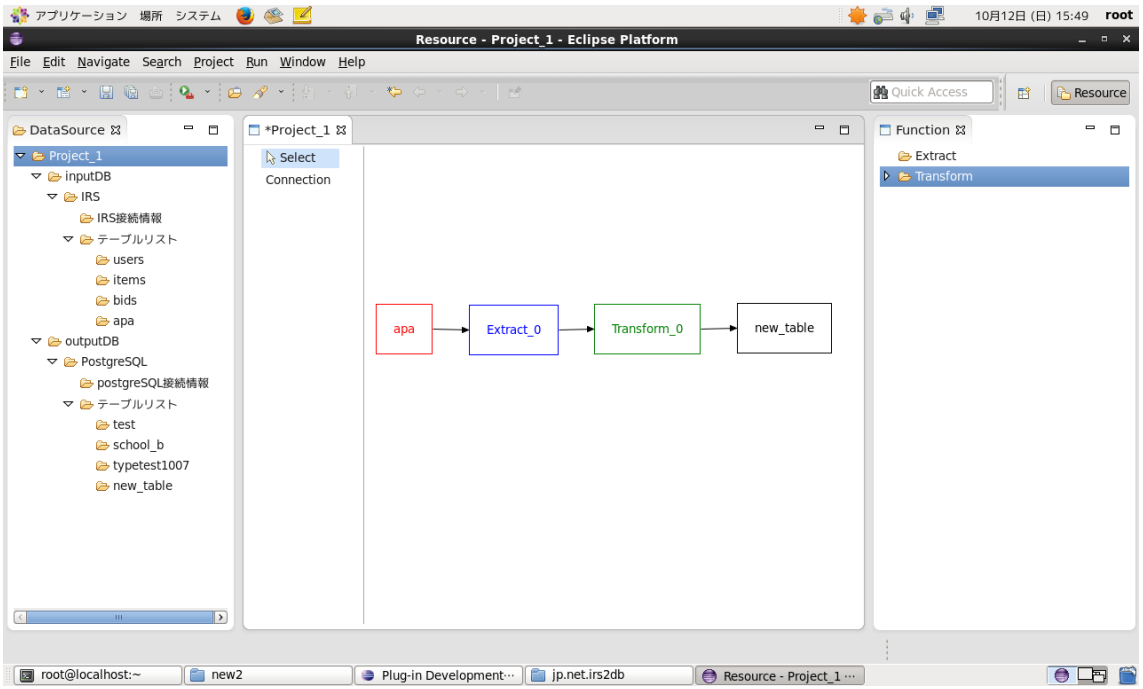
結果：該当アイコンが削除される。



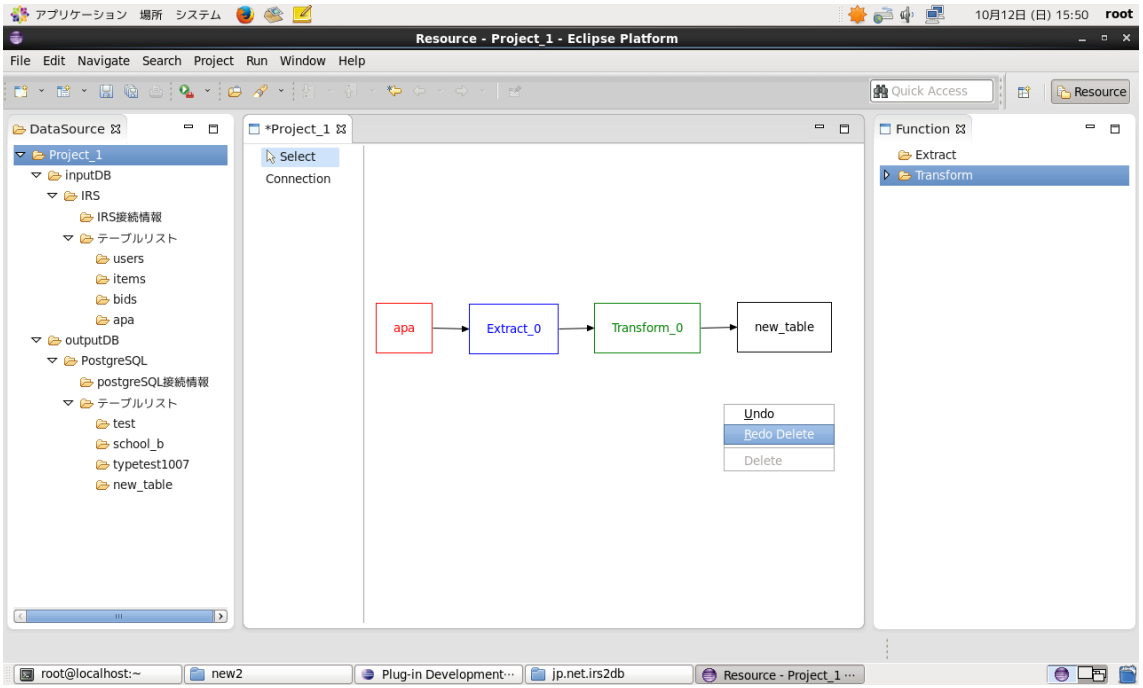
操作５：作業ウィンドウを右クリックして、「Undo」を選択する。



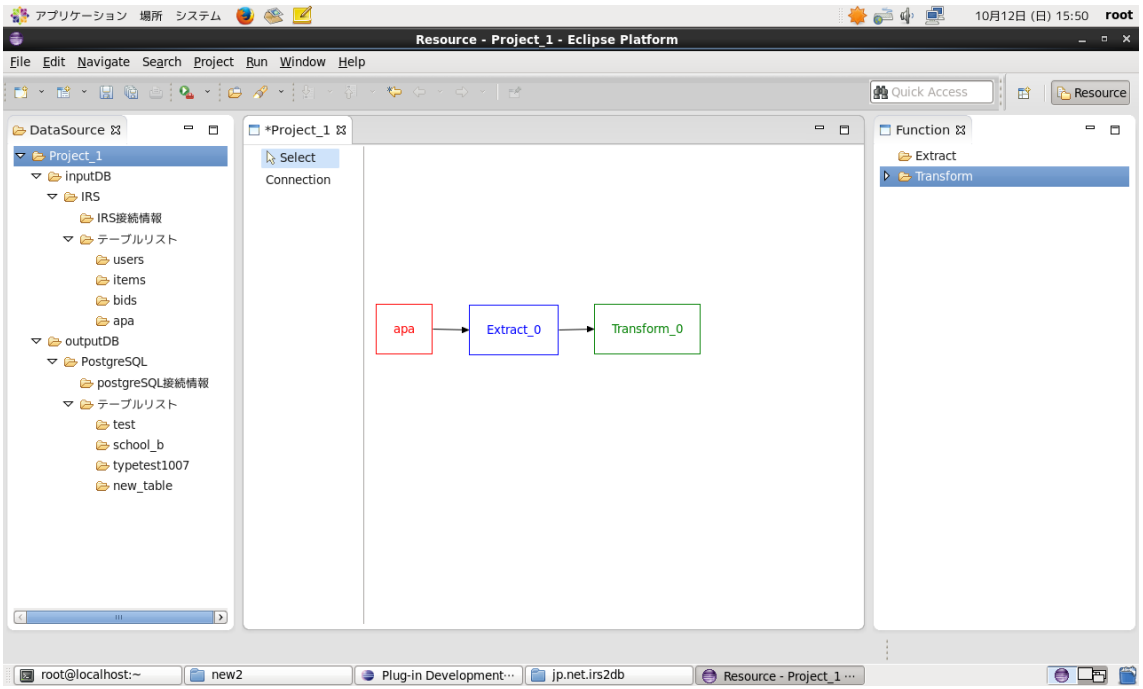
結果：削除したアイコンが復元される。



操作 6：作業ウィンドウを右クリックして、「Redo」を選択する。



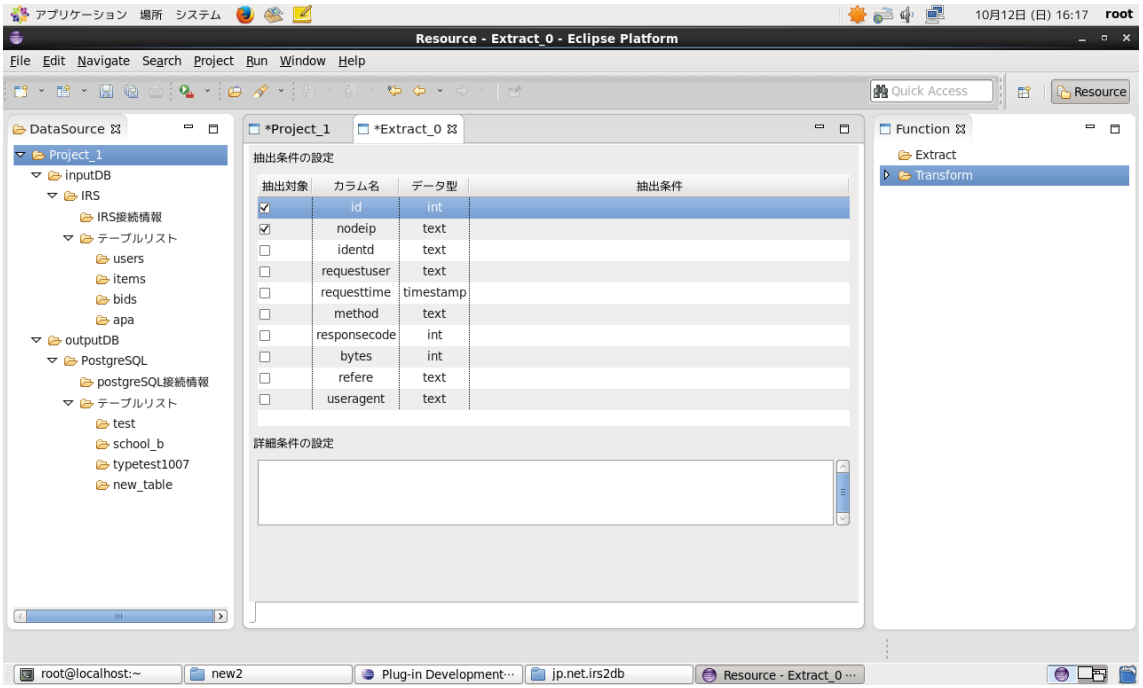
結果：回復されたアイコンはもう一度削除されている。



1.6. Extract 作業の編集

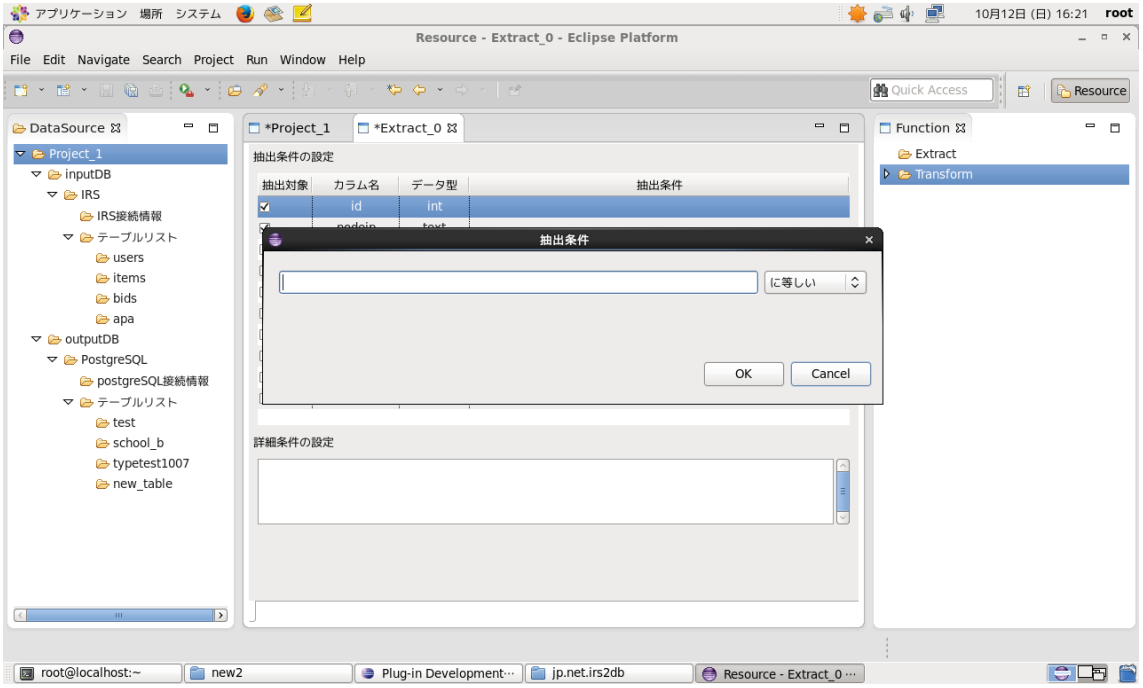
操作 1：Extract アイコンをダブルクリックする。

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

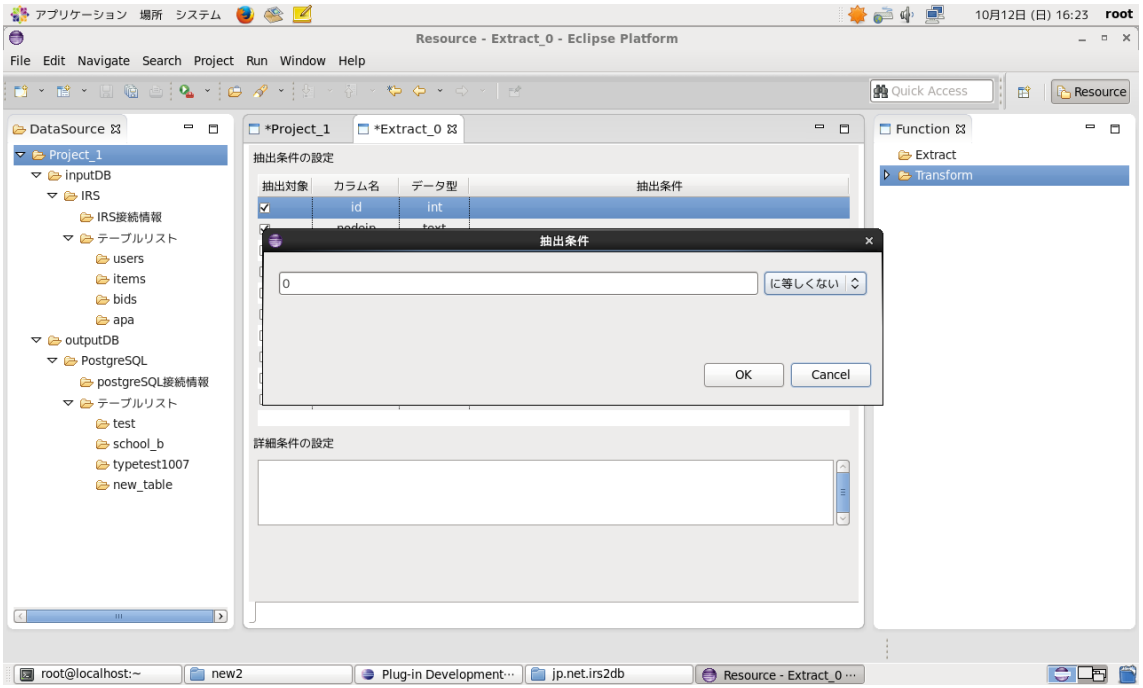


操作２：カラムの「抽出条件」をクリックします。

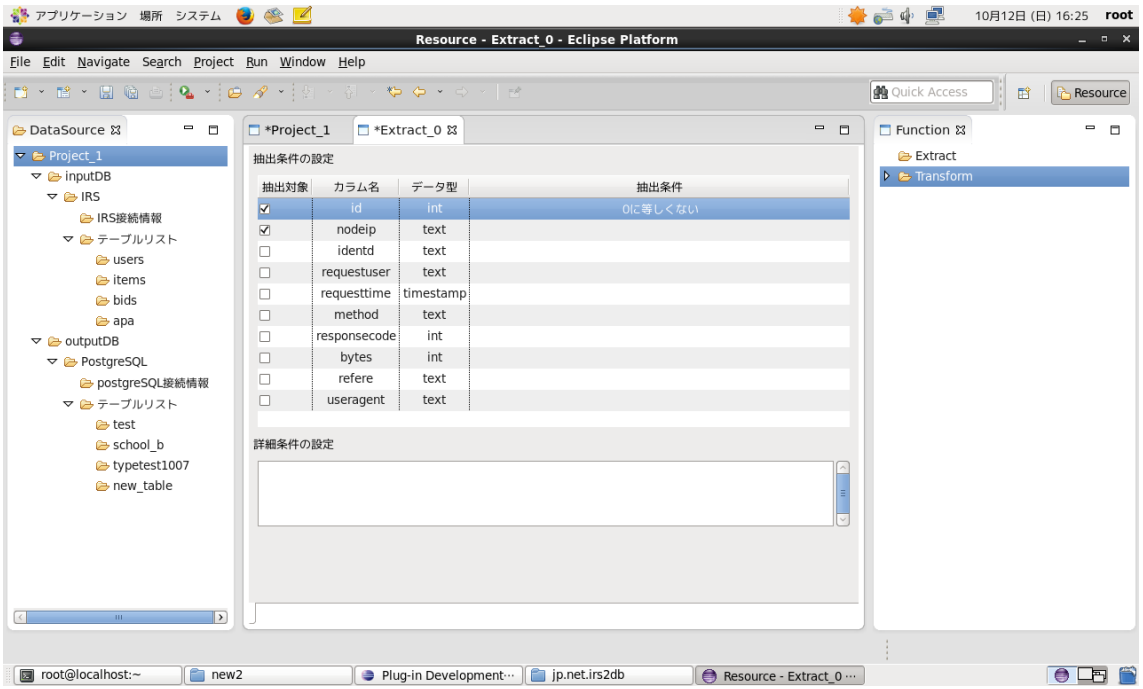
結果：抽出条件編集画面が表示される。



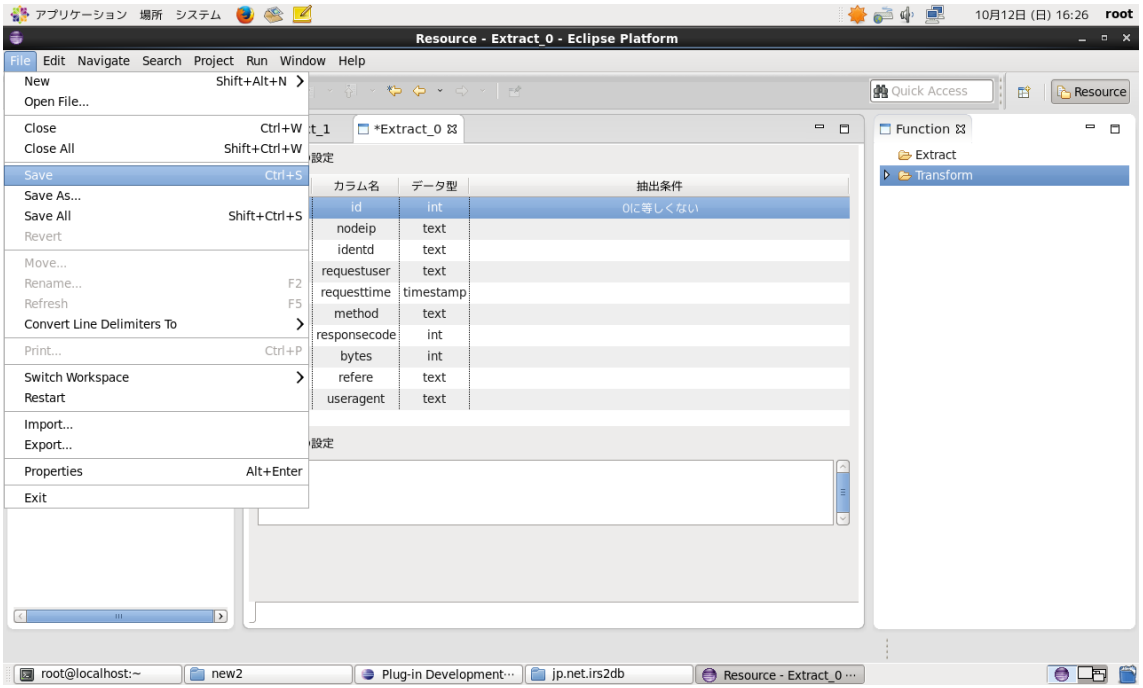
操作３：抽出条件を入力して、「OK」ボタンを押す。



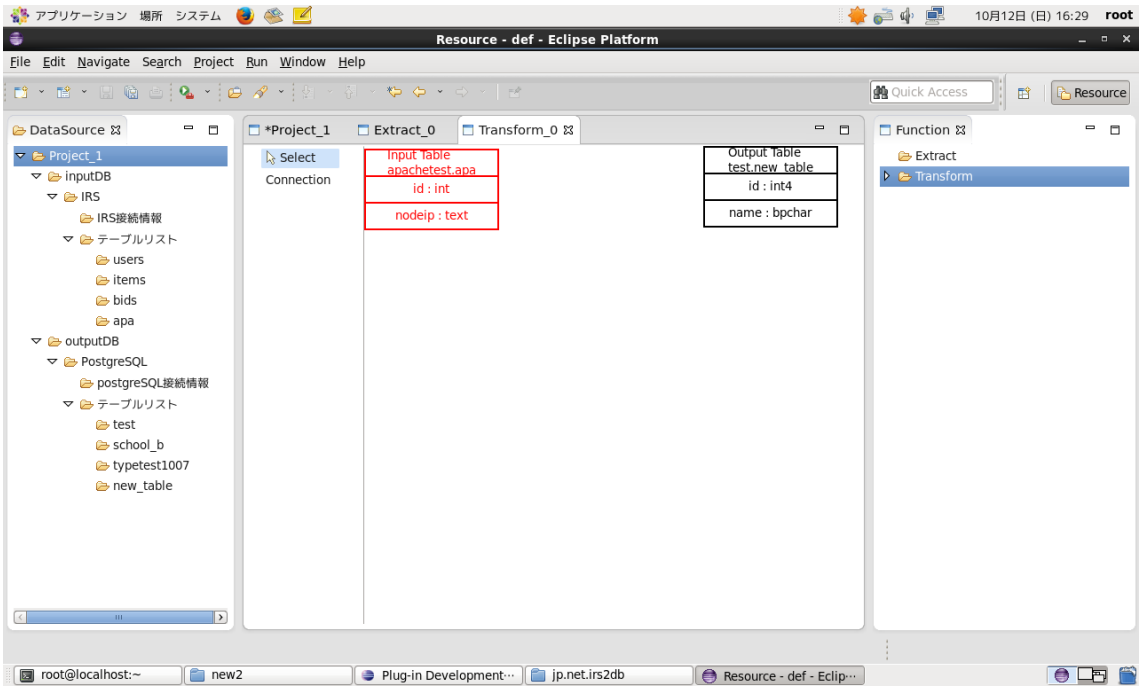
結果：Extract 編集画面に該当カラムの抽出条件が表示される。



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する。



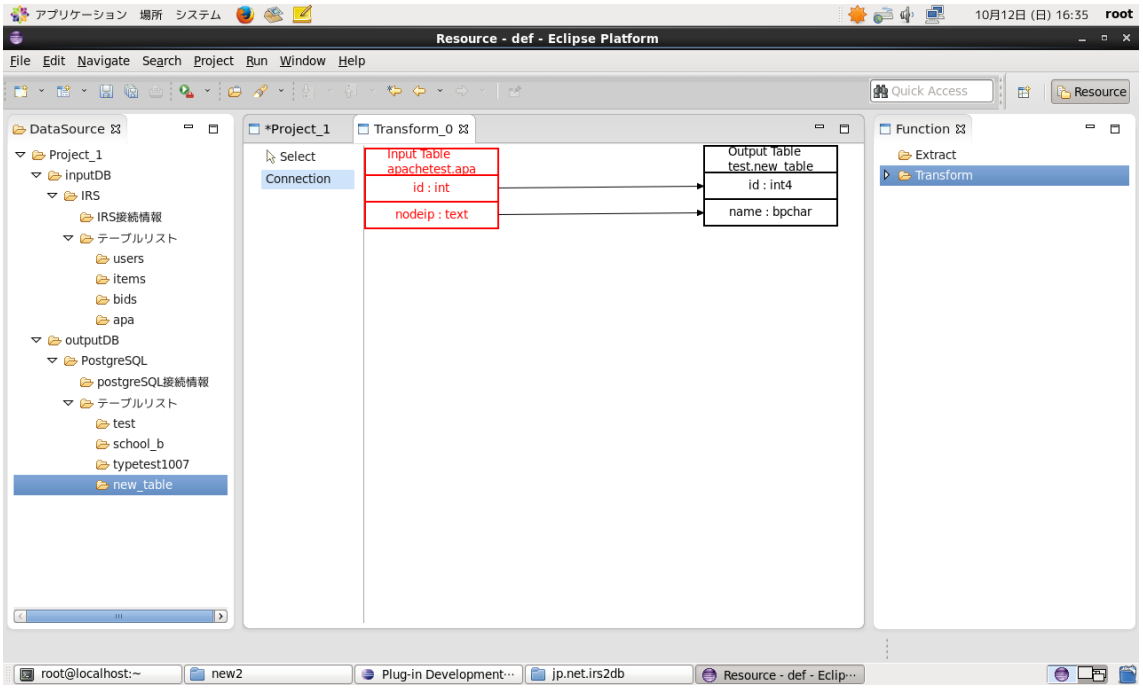
結果：Transform 編集画面に抽出後 Input Table が表示されている。



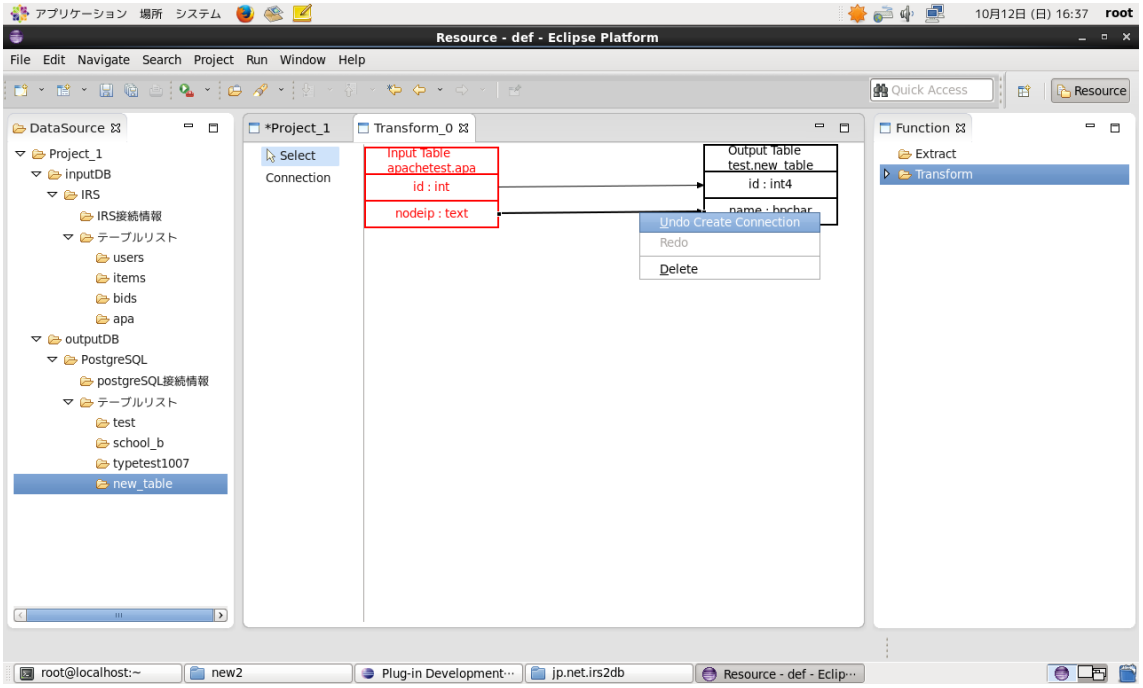
1.7. Transform 作業の編集

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する。

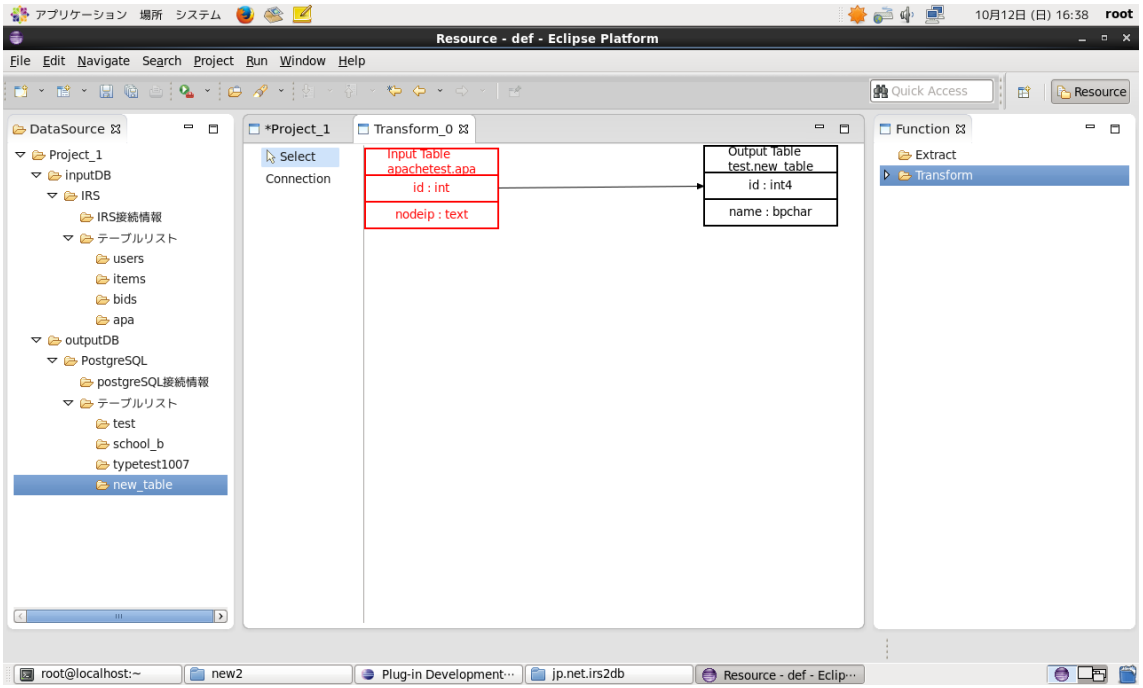
結果：対応関係を指定している線が表示されている。



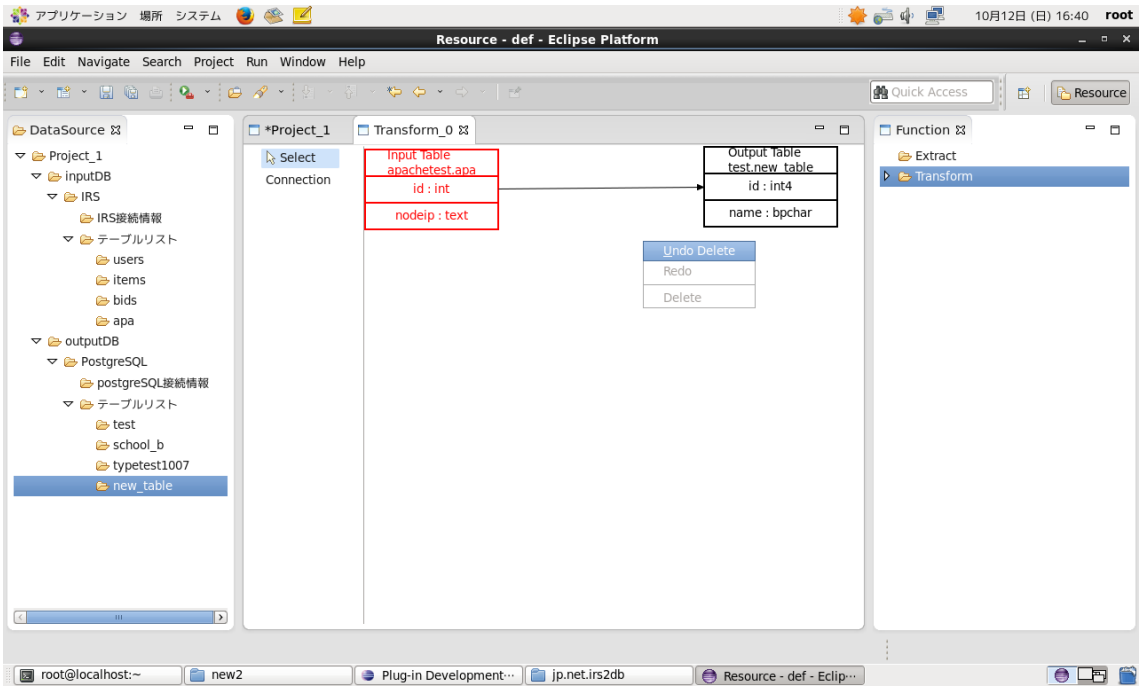
操作２：線を右クリックして、「Delete」を選択する。



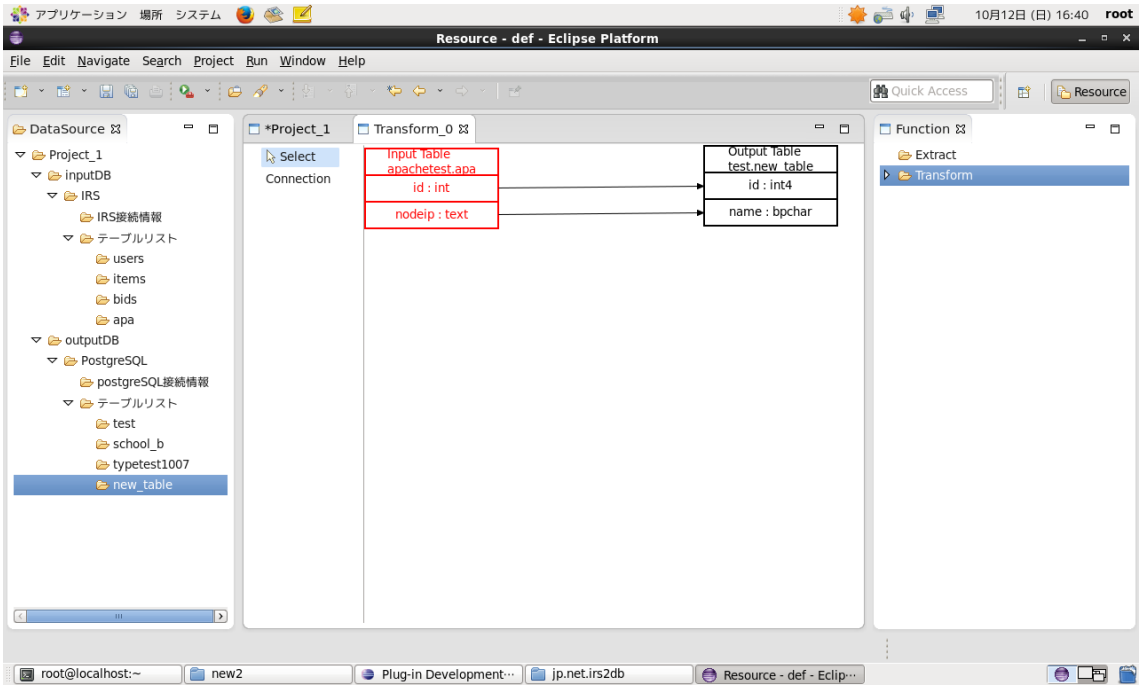
結果：選択されている線が消される。



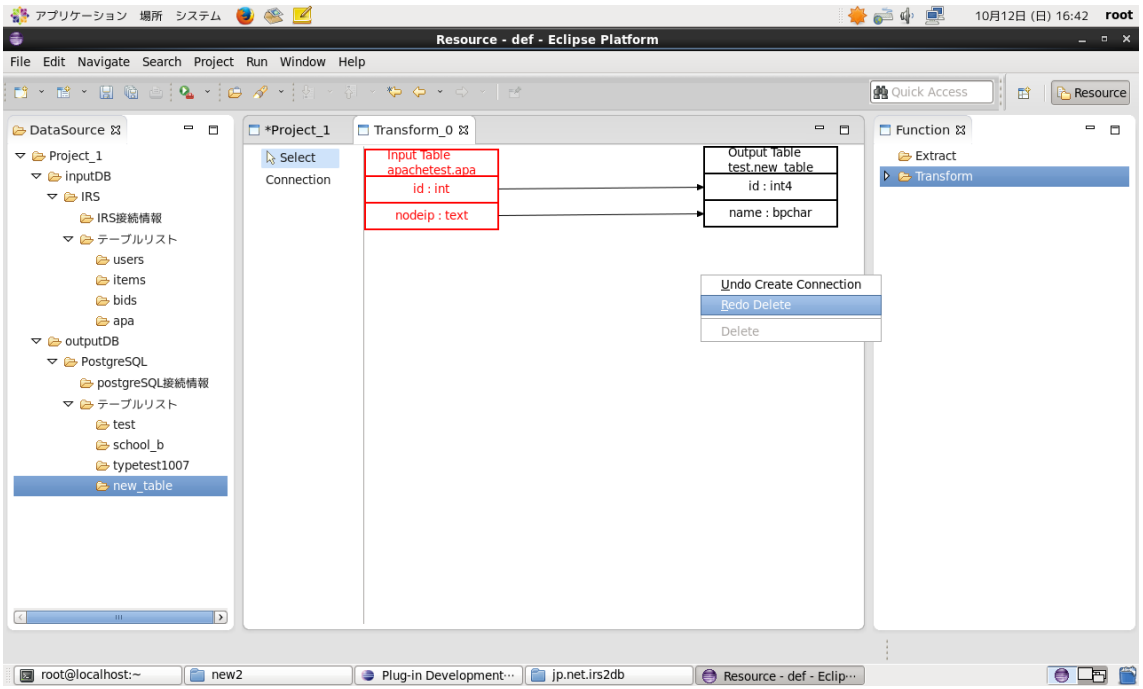
操作 3：Transform 編集画面を右クリックして、「Undo」を選択する。



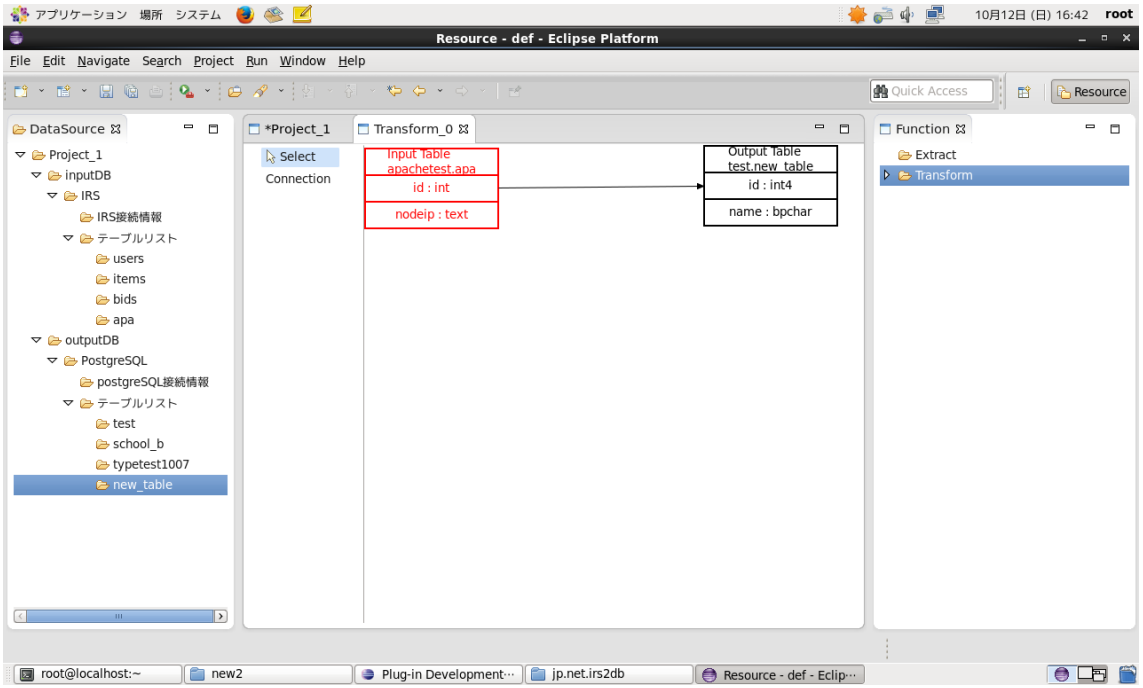
結果：削除された線が復元される。



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



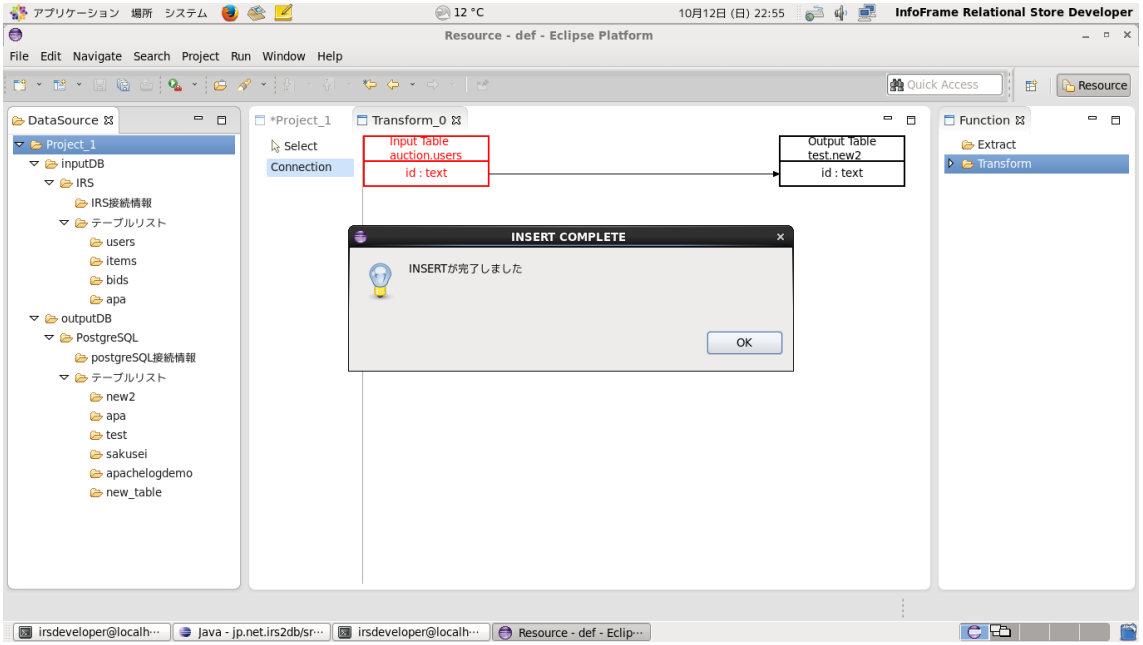
結果：回復された線がもう一度消される。



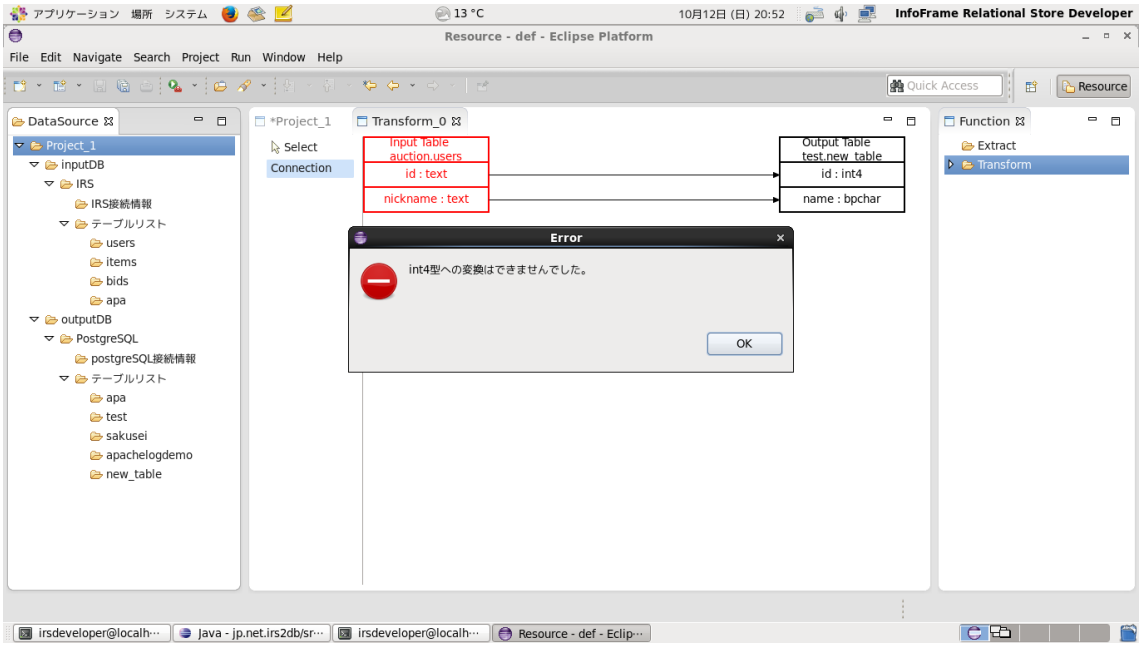
1.8. ETL の実行

操作：プロジェクト名を右クリックして、「実行」を選択する。

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る。



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る。



以上のように、「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認した。

第５節　　第２イテレーション総合テスト報告書

本節では、実施した総合テストについて報告する。

第１項　　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る。

第２項　　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表 3 に示す。

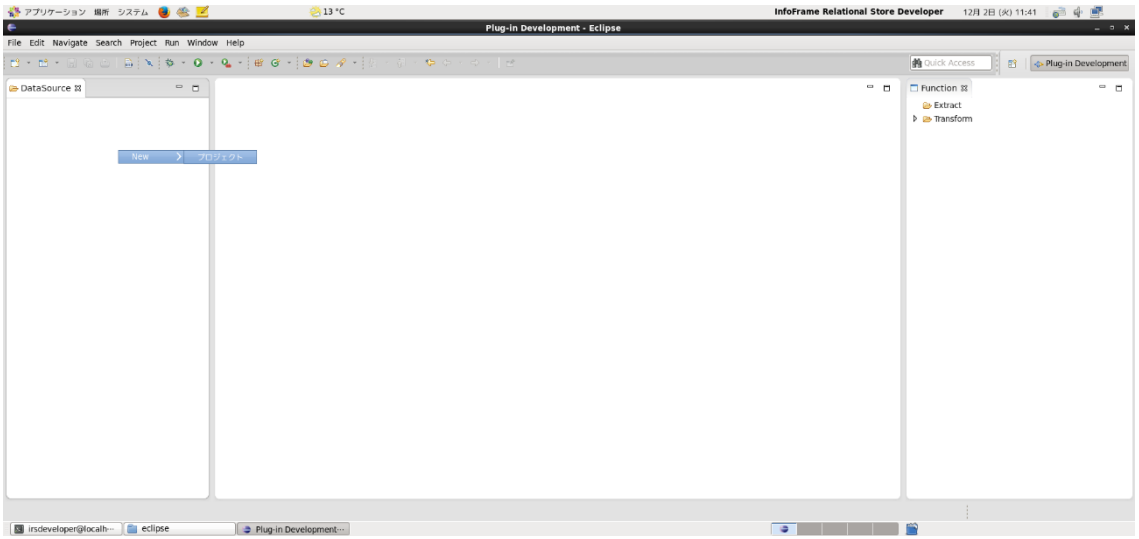
表 3　テスト環境

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

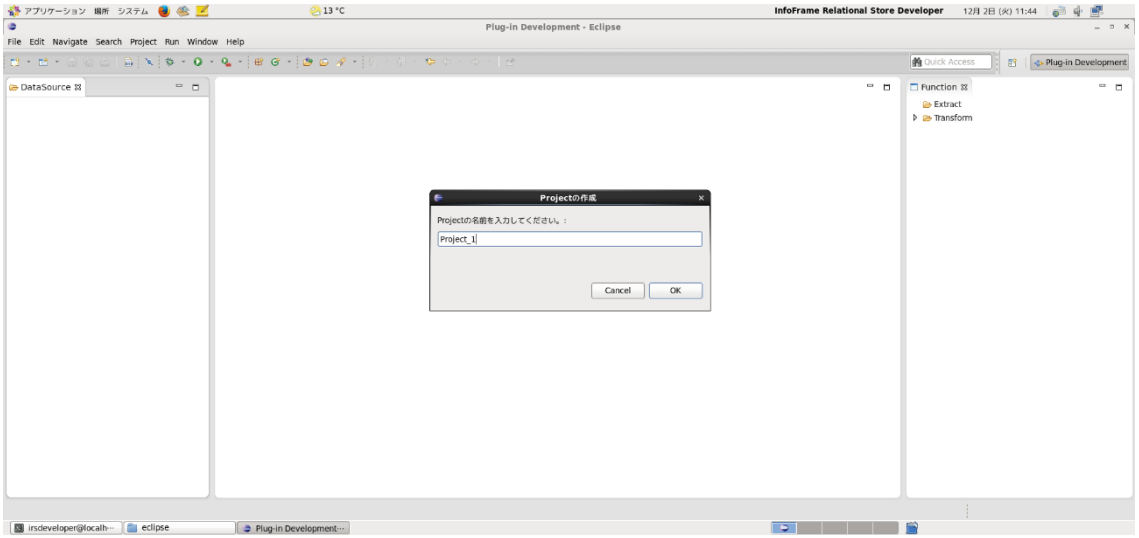
第３項　　テスト操作と結果

2.　プロジェクト作成

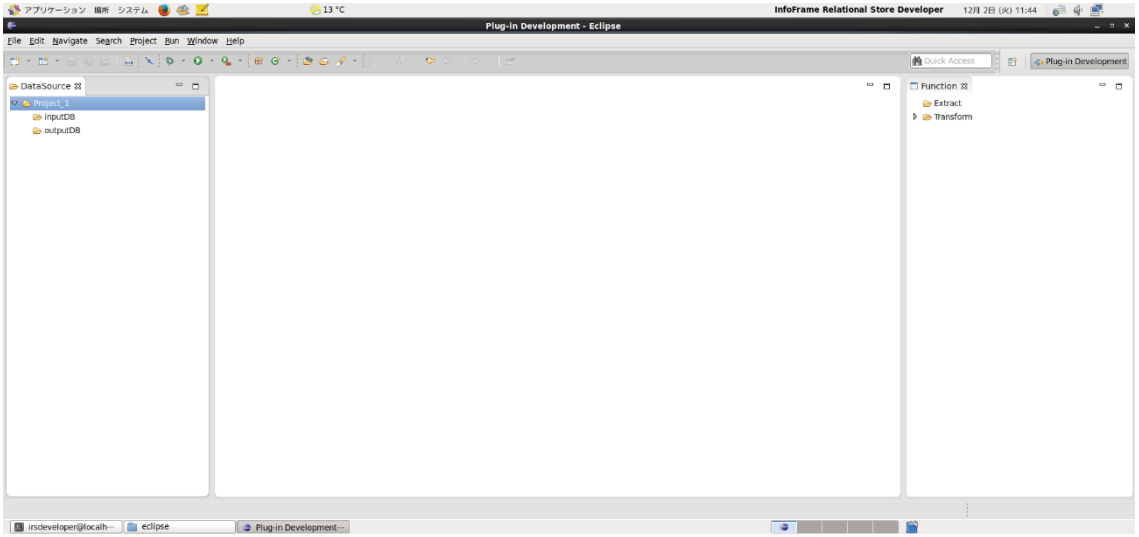
操作１：DataSource ビューで右クリックして、プロジェクトを選択する。



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。

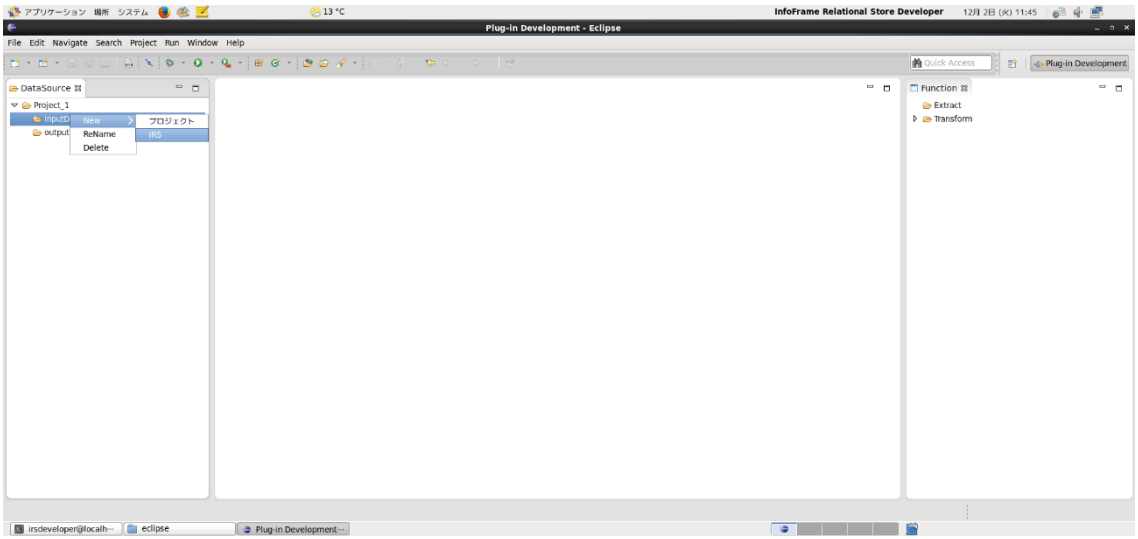


結果：「Project_1」というプロジェクトが新規作成され、「inputDB」と「outputDB」が表示される。

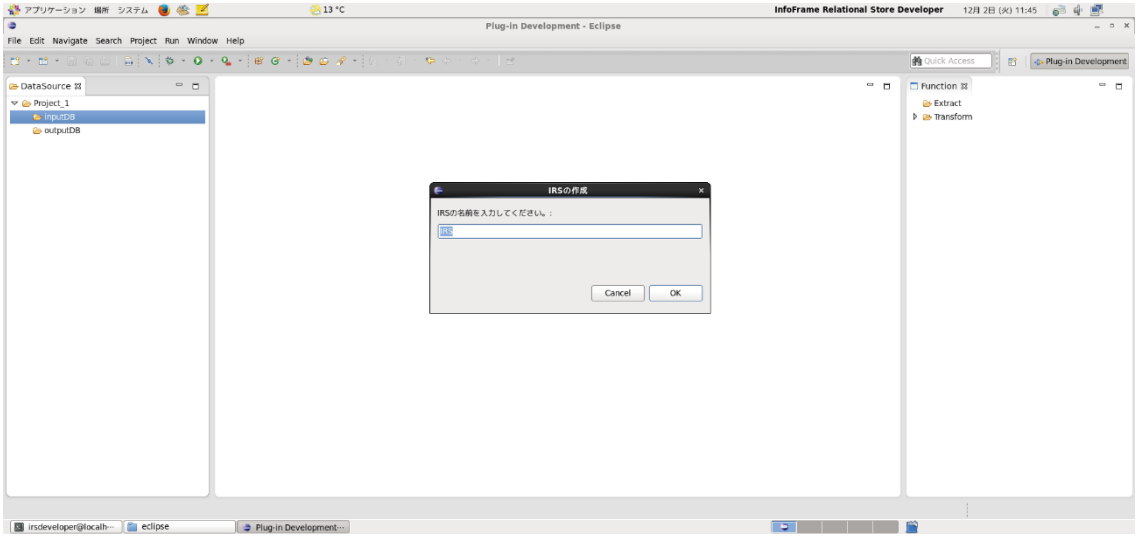


2.1. IRS 接続

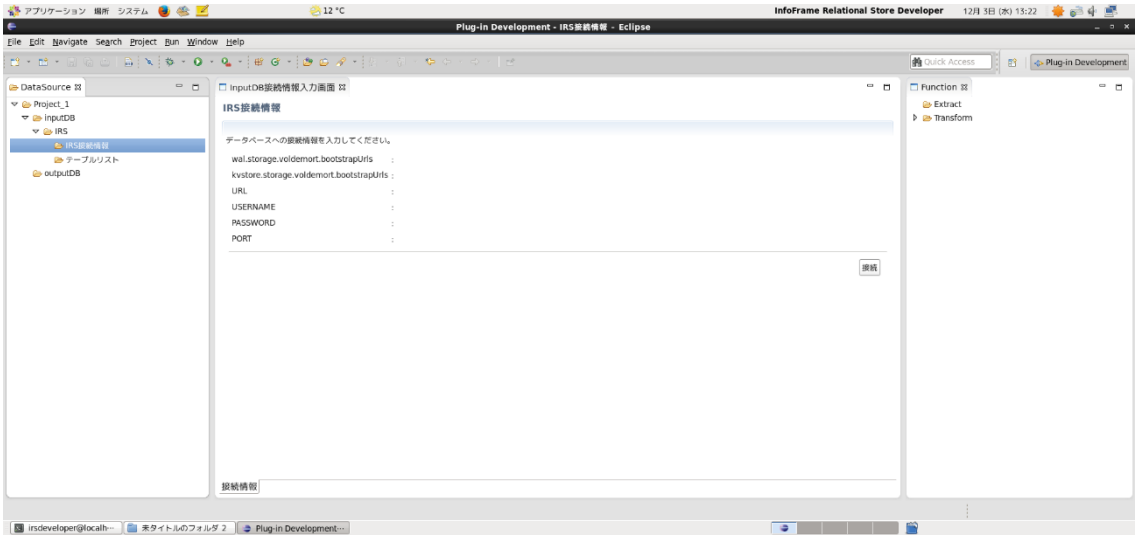
操作 1：「inputDB」を右クリックして、「New」→「IRS」を選択する。



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



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

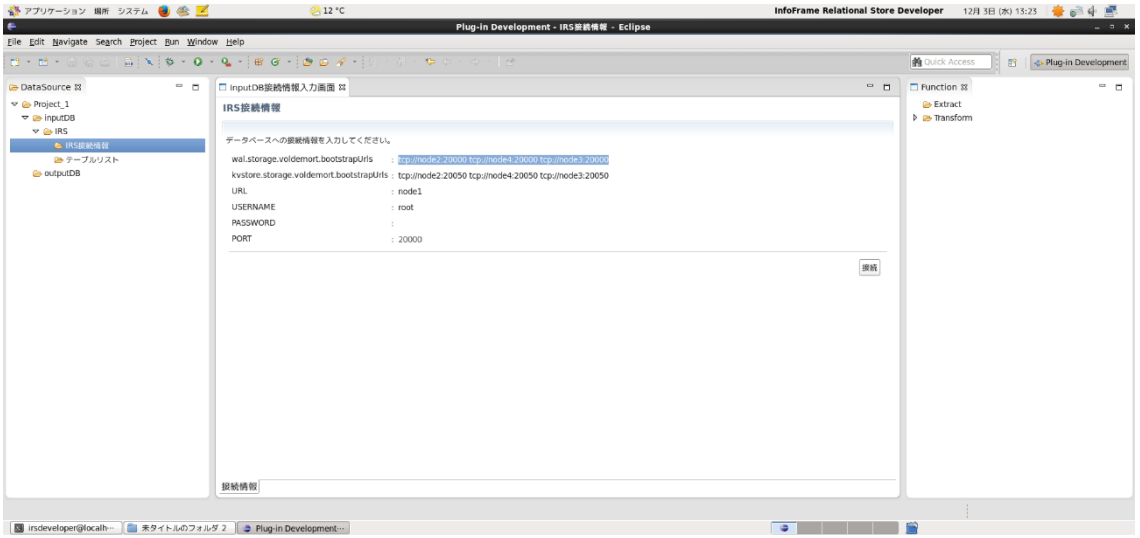


※　１件のバグ報告

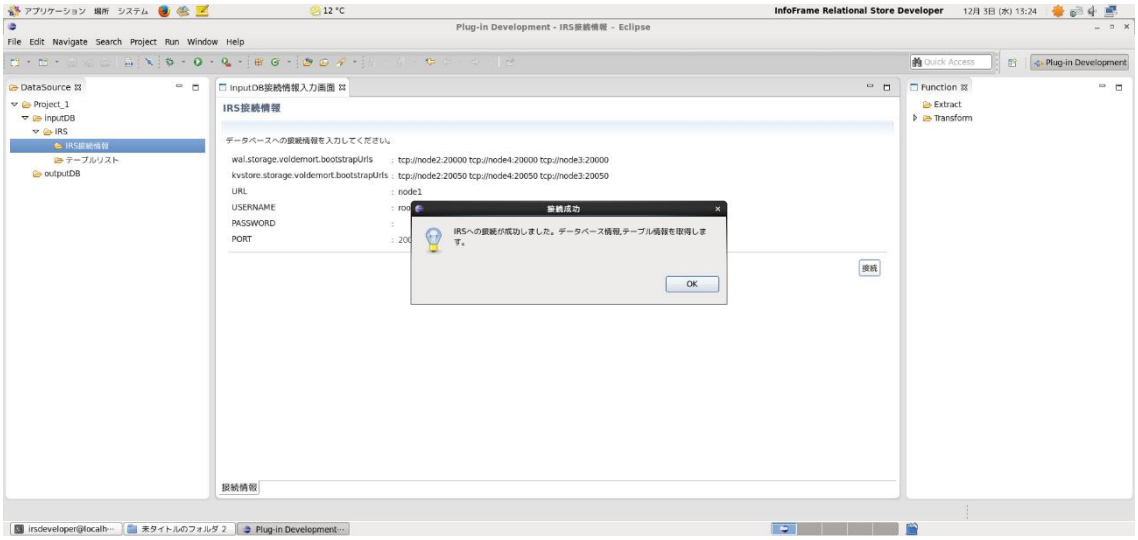
バグ:2014/12/3 12:00 バグ PASSWORD が「PASSWORED」となっている。

対処:PASSWORD とする。

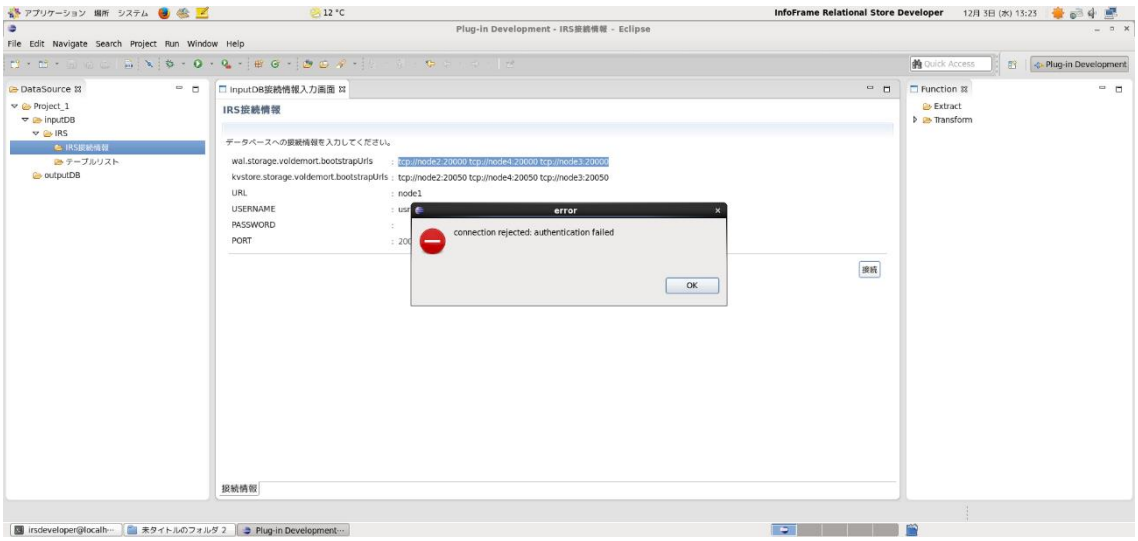
操作４:「InputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押す。



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る。

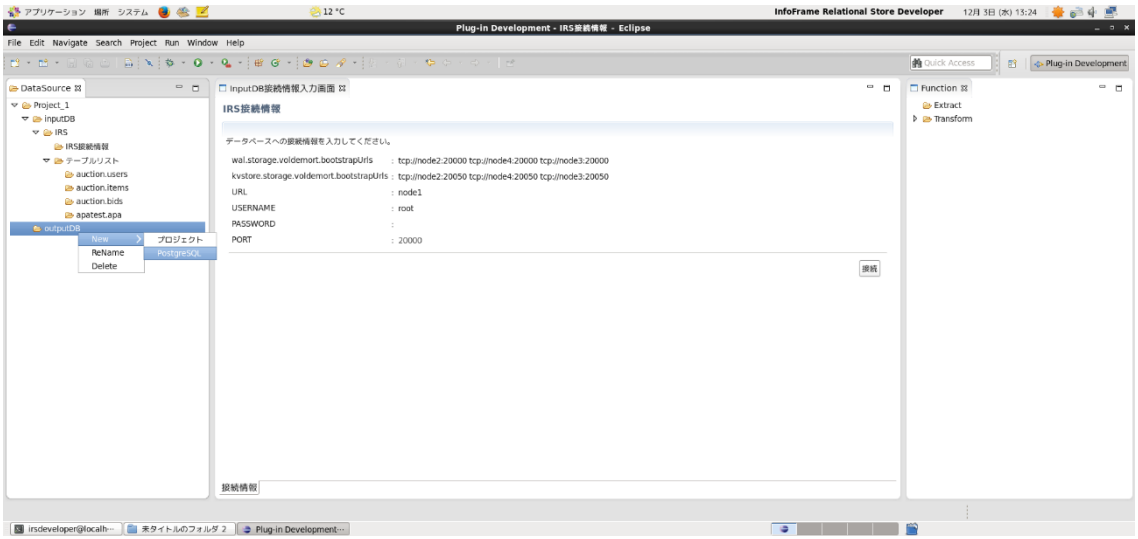


結果 B：接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示される。

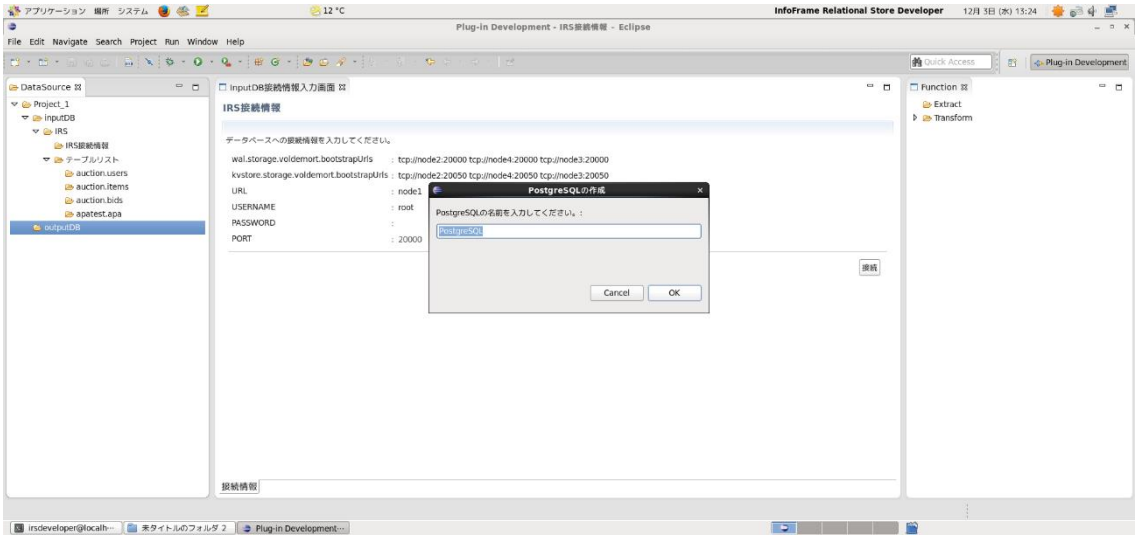


2.2. PostgreSQL 接続

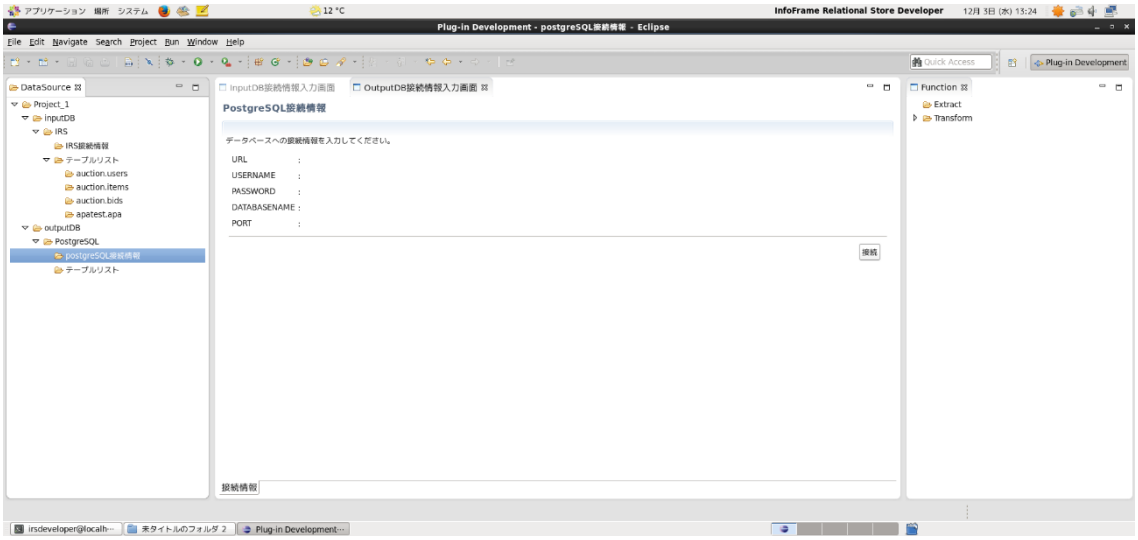
操作 1：「outputDB」を右クリックして、「New」→「PostgreSQL」を選択する。



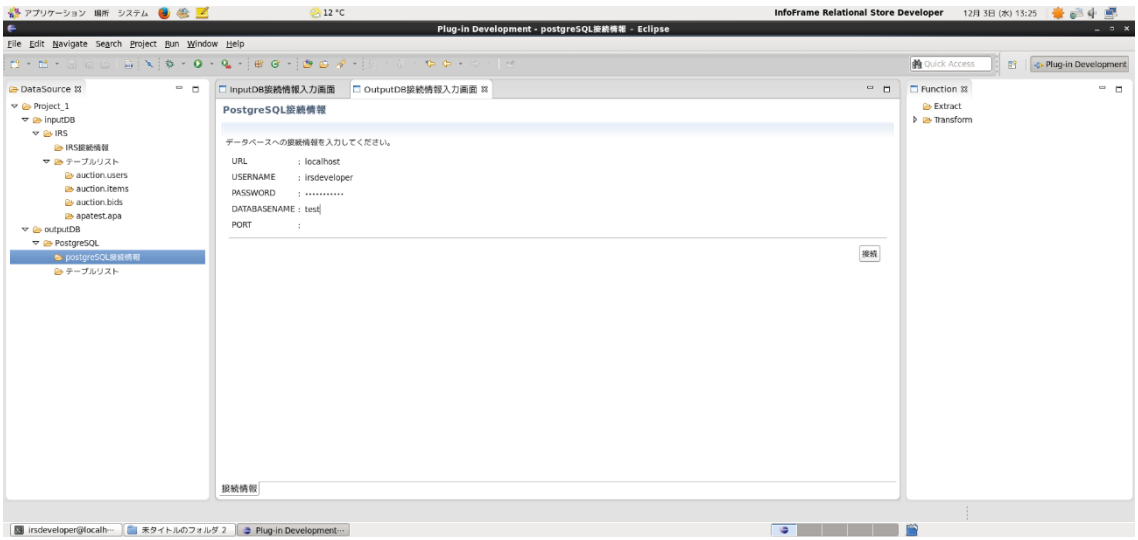
操作 2：「PostgreSQL 作成画面」で PostgreSQL の名前を入力して、「OK」ボタンを押す。



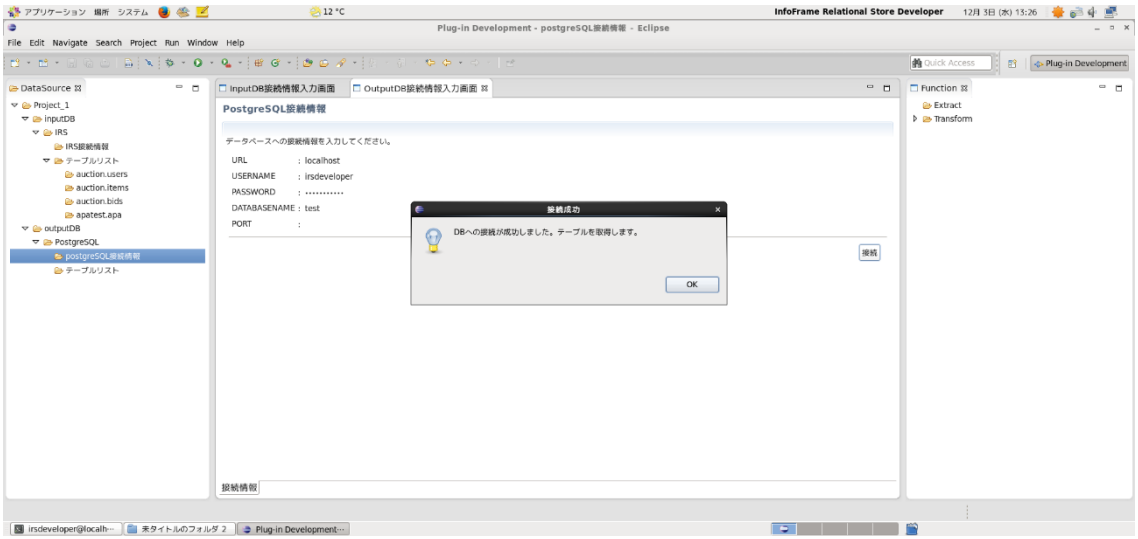
操作 3：作成した「PostgreSQL」を選択し、「postgreSQL 接続情報」をダブルクリックする。



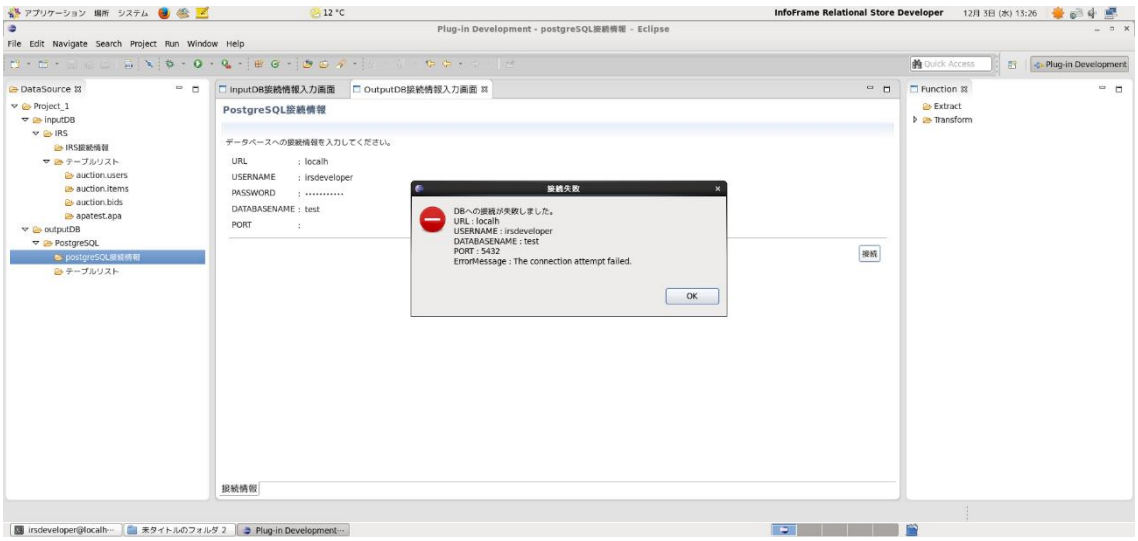
操作 4：「OutputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押す。



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る。

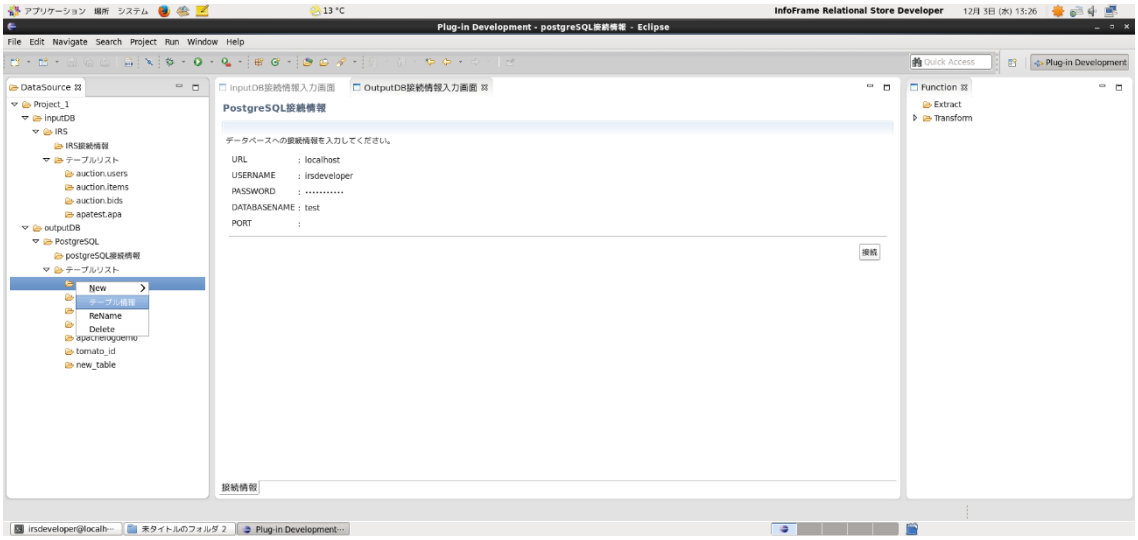


結果 B：接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示される。

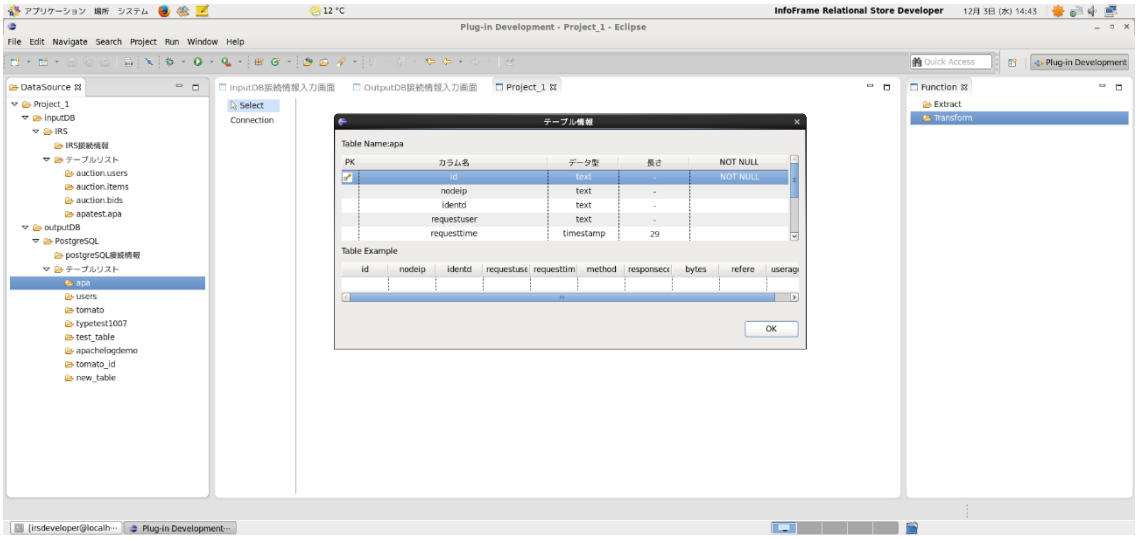


2.3. テーブル情報の表示

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」を選択する。

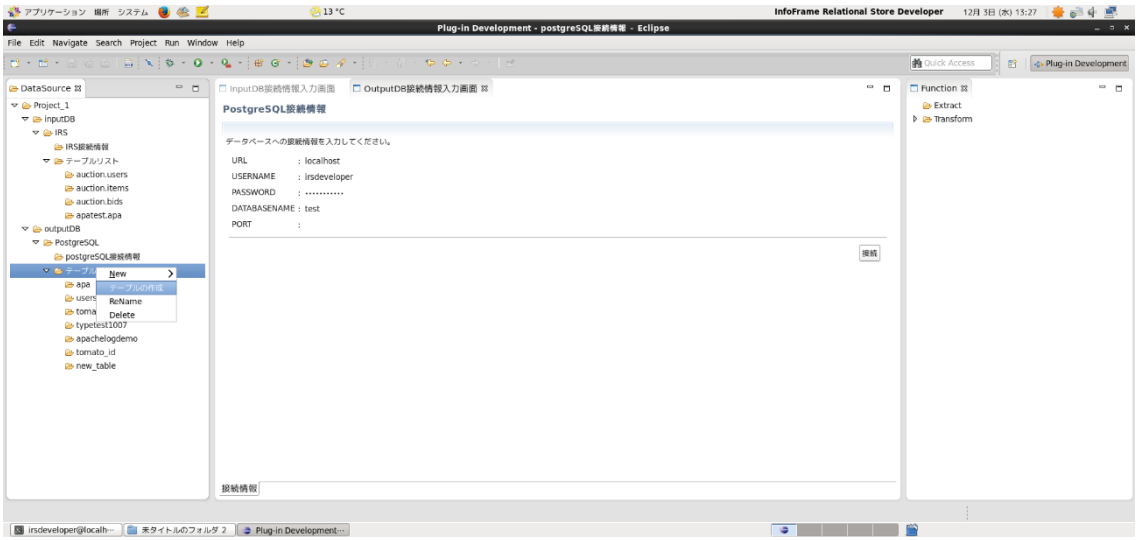


結果：テーブル情報が表示される。TableExample はデータが元テーブルに存在しない場合は空欄で表示する。

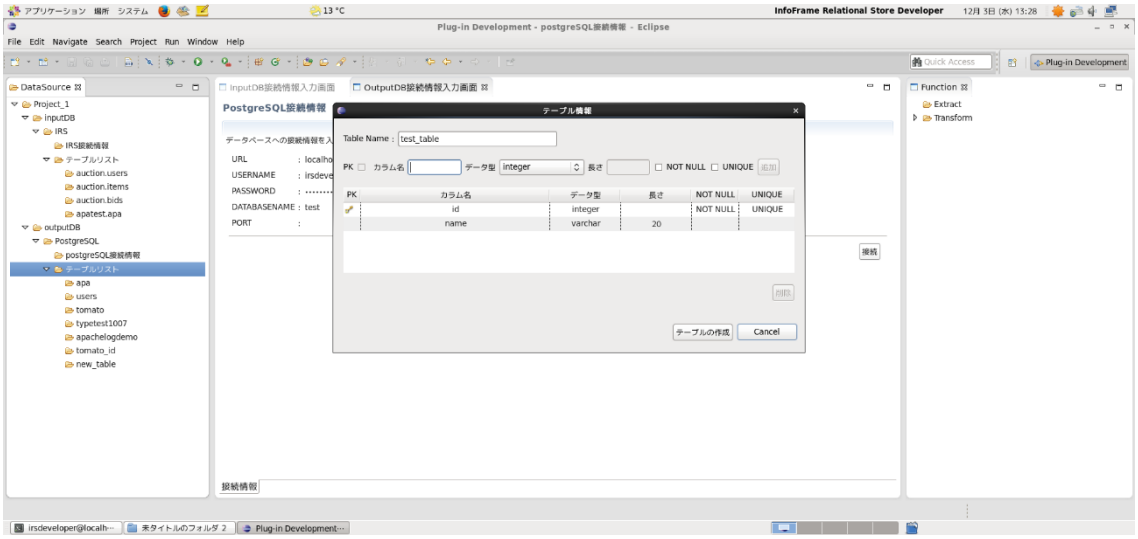


2.4. PostgreSQL に新規テーブルを作成

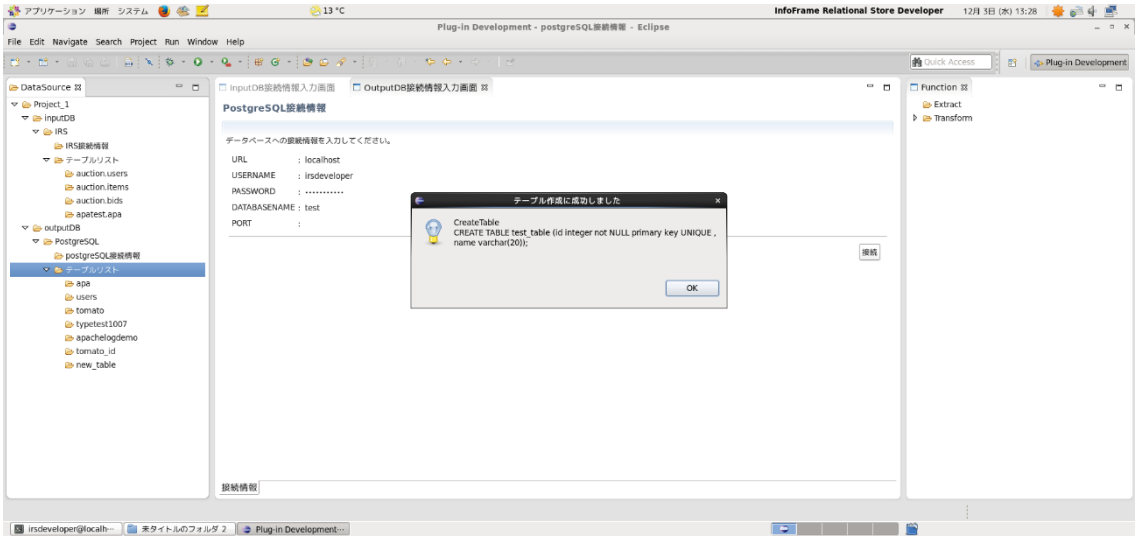
操作１：「テーブルリスト」を右クリックして、「テーブルの作成」を選択する。



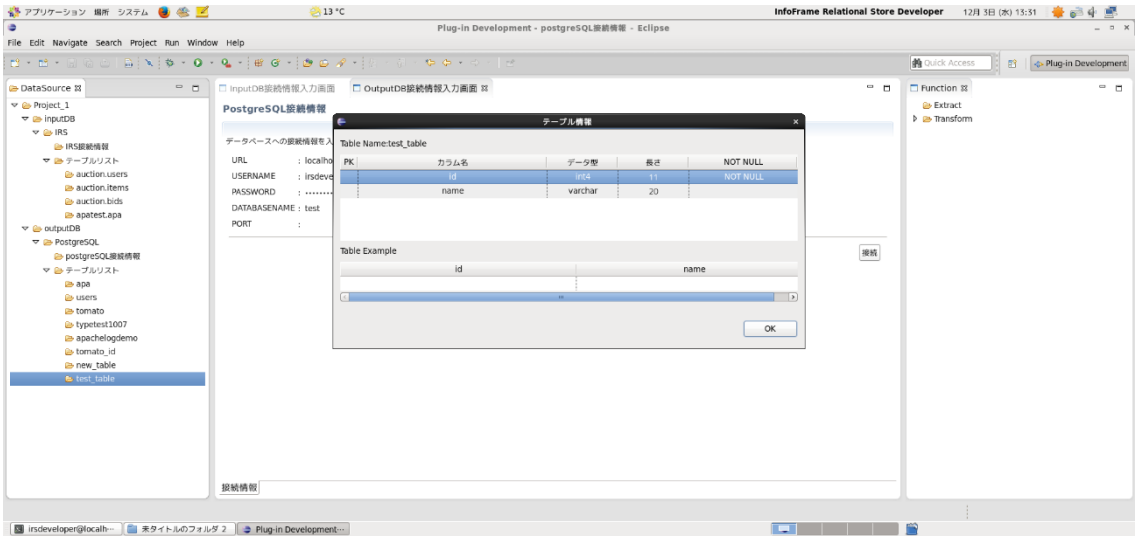
操作２：「テーブル情報」で作成するテーブルの情報を入力して、「テーブルの作成」ボタンを押す。



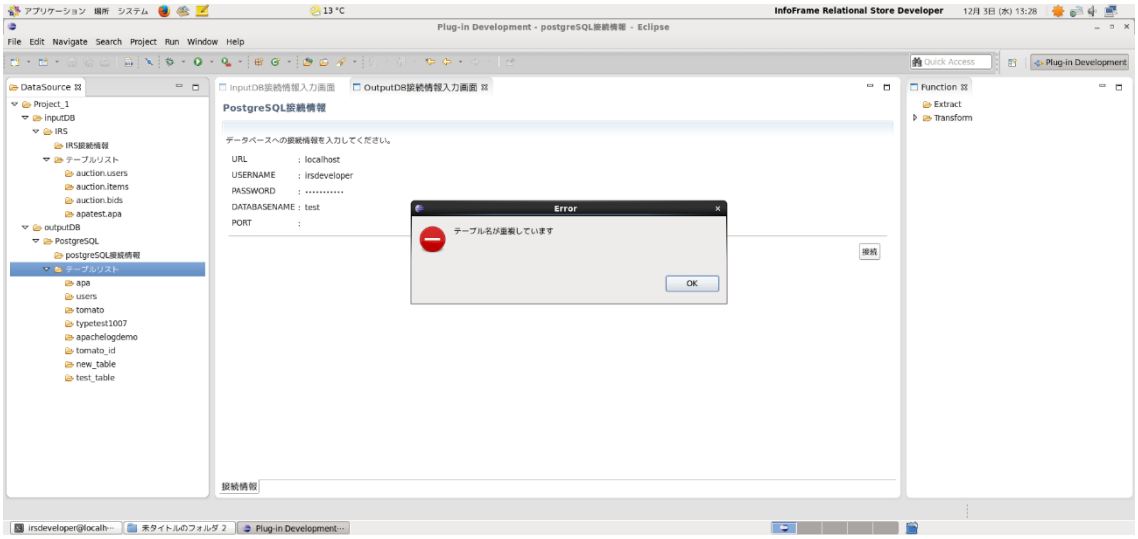
結果 A-1：正しく作成できた場合、「テーブルの作成に成功しました」ポップアップが表示されている。OK ボタンを押すと、テーブルリストにテーブル名が追加される。



結果 A-2：適切に作られていることを確認できる

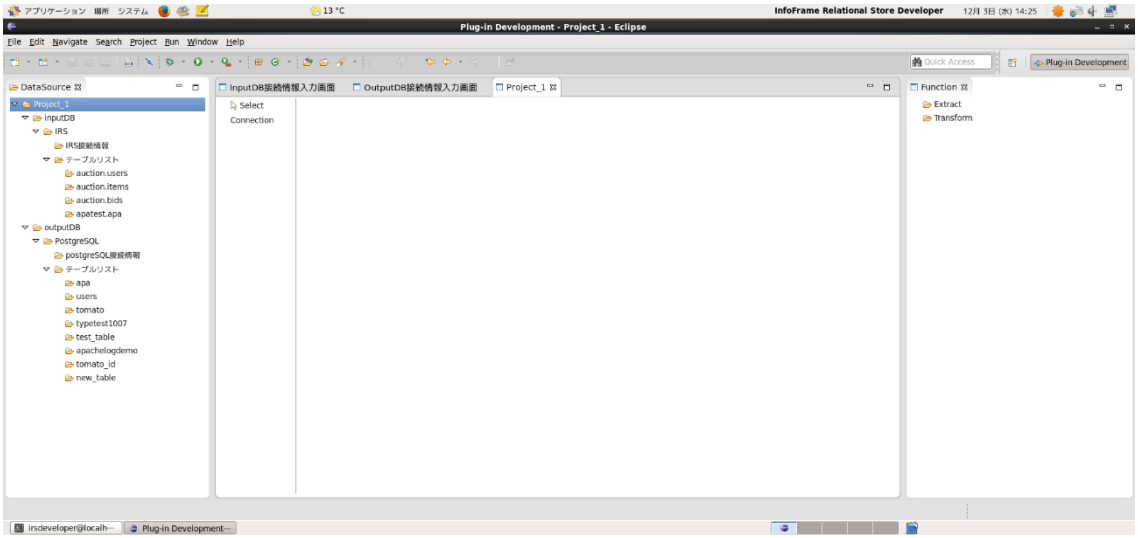


結果 B：デーブル作成が失敗した場合は、エラー画面が表示される。



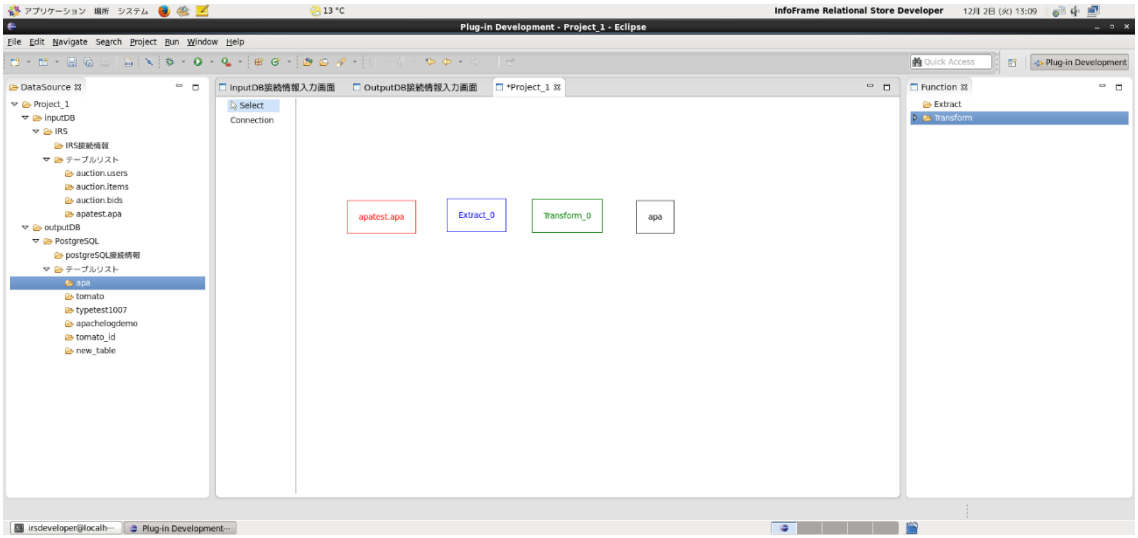
2.5. ETL 作業の編集

操作 1：プロジェクト名をダブルクリックすることで、作業ウィンドウが開く。



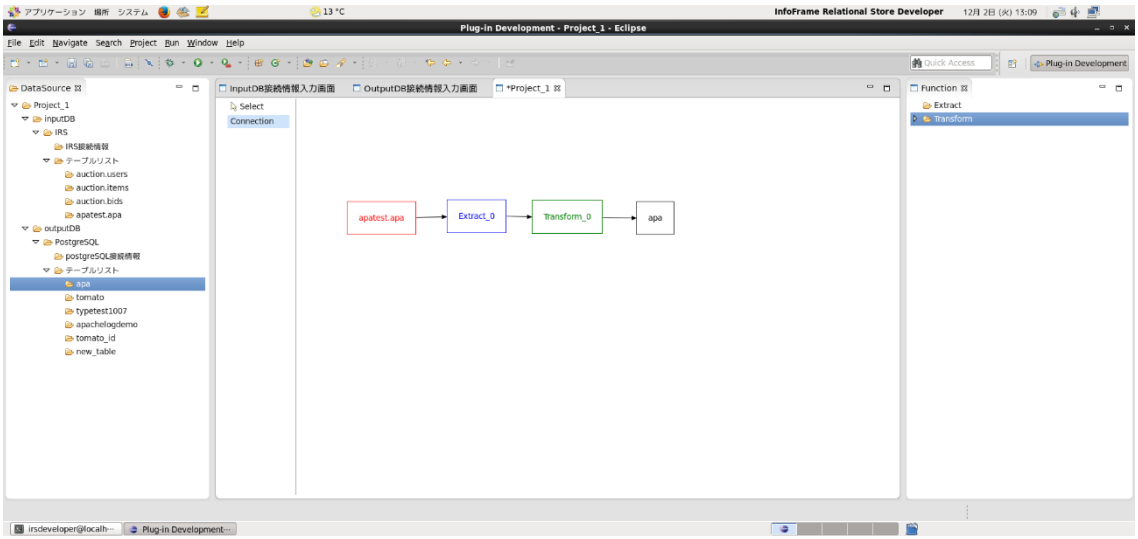
操作 2:使用するテーブルやファンクションなどを作業ウィンドウにドラッグする。

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る。

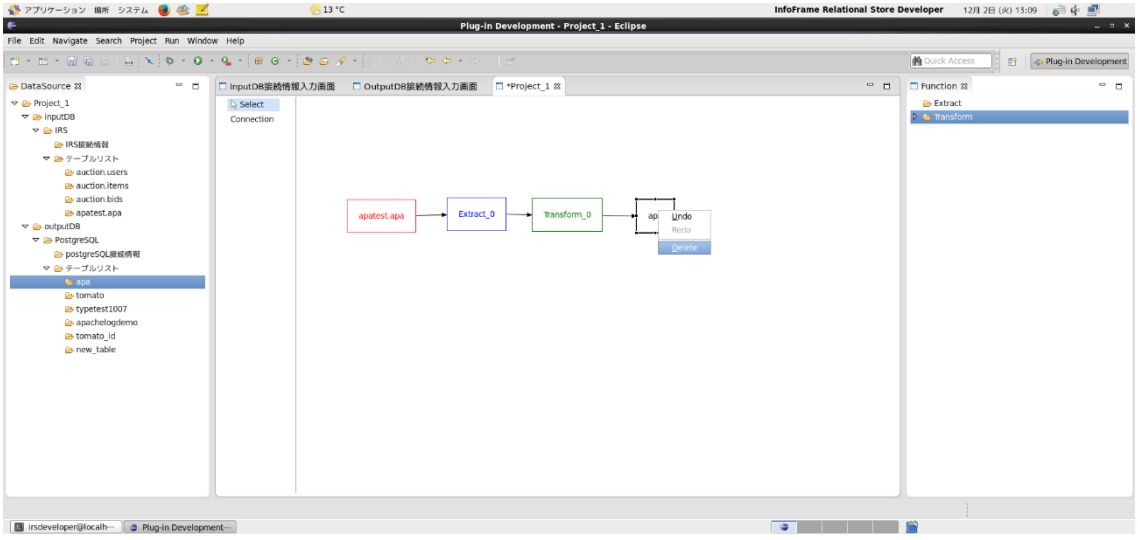


操作 3 : 左側の「Connection」を選択して、作業ウィンドウ内のアイコンを紐付けする。

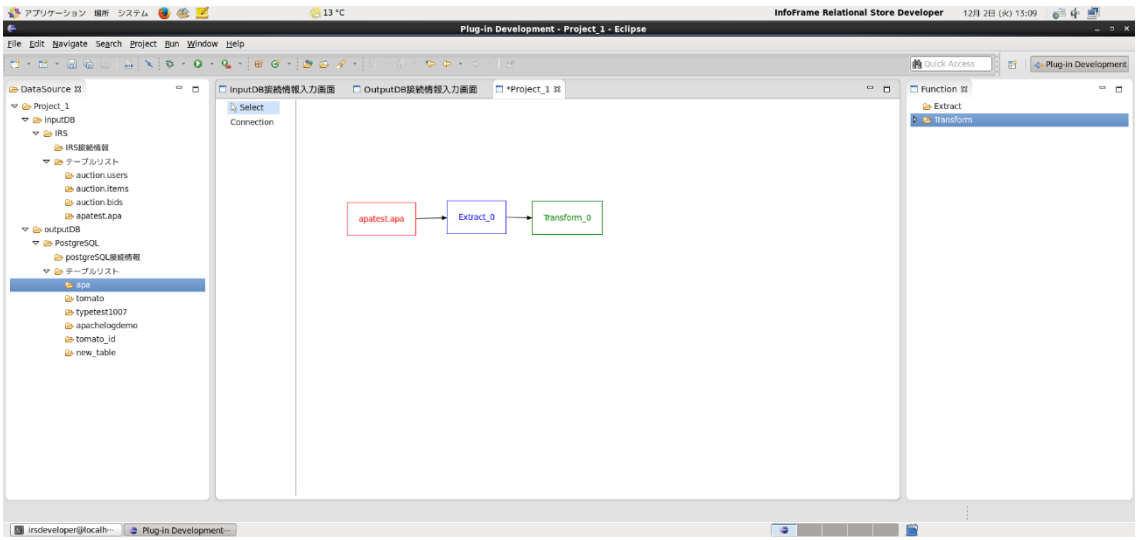
結果：テーブルと各ファンクションの対応関係が指定されている。



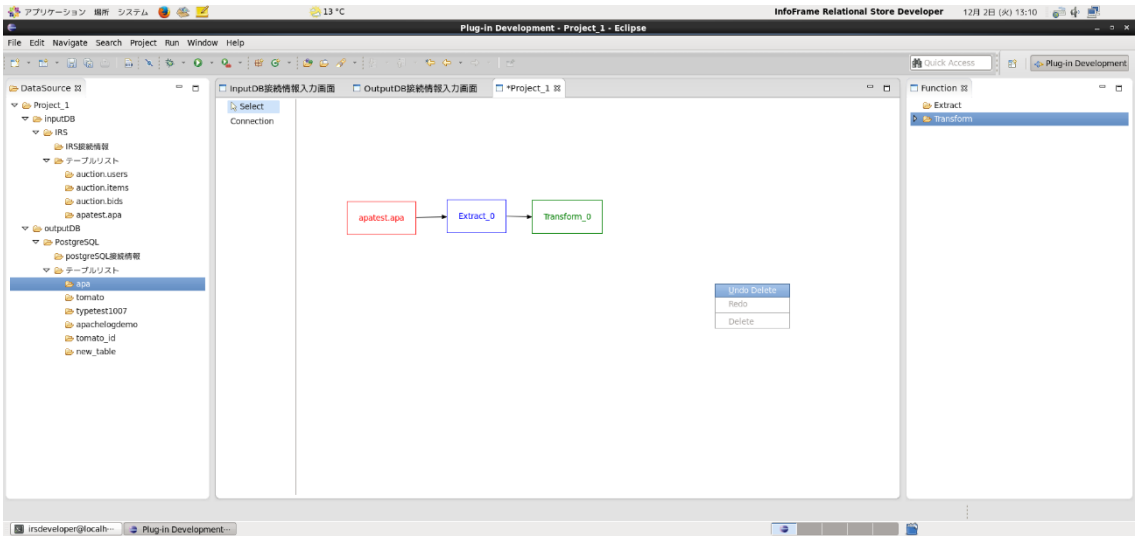
操作４：アイコンを右クリックして、「Delete」を選択する。



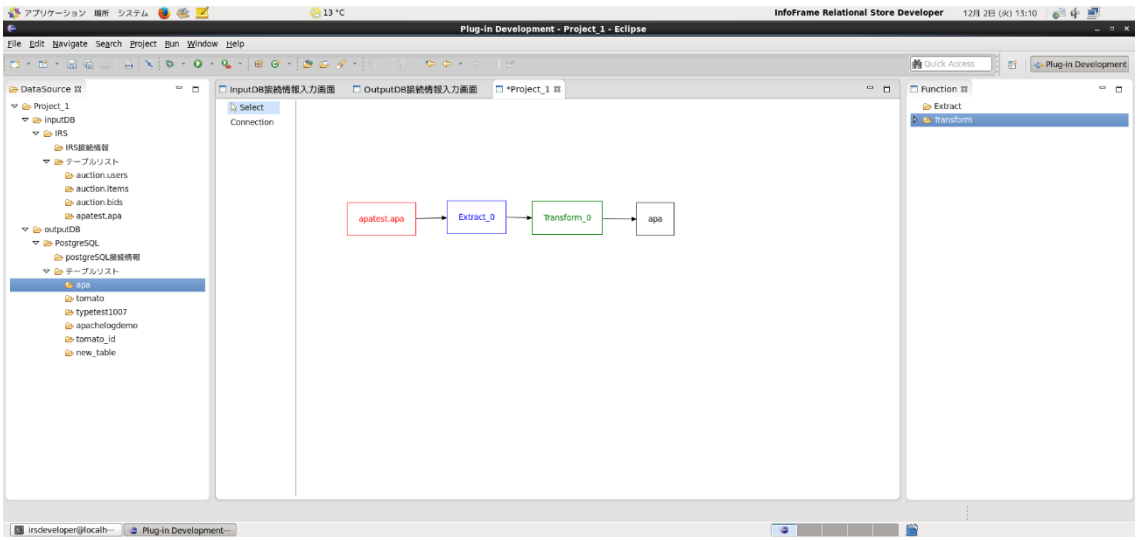
結果：該当アイコンが削除される。



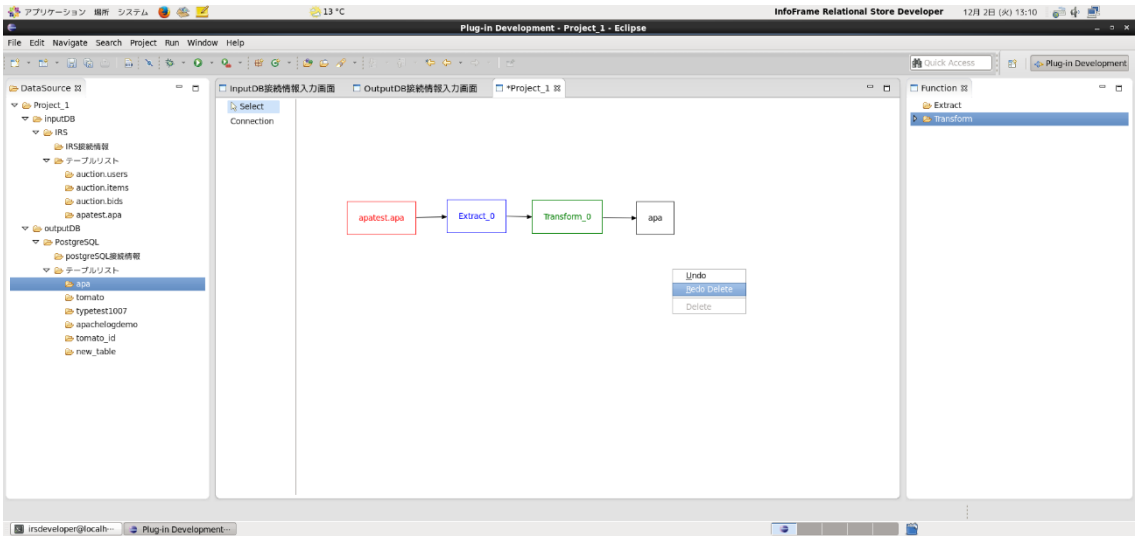
操作５：作業ウィンドウを右クリックして、「Undo」を選択する。



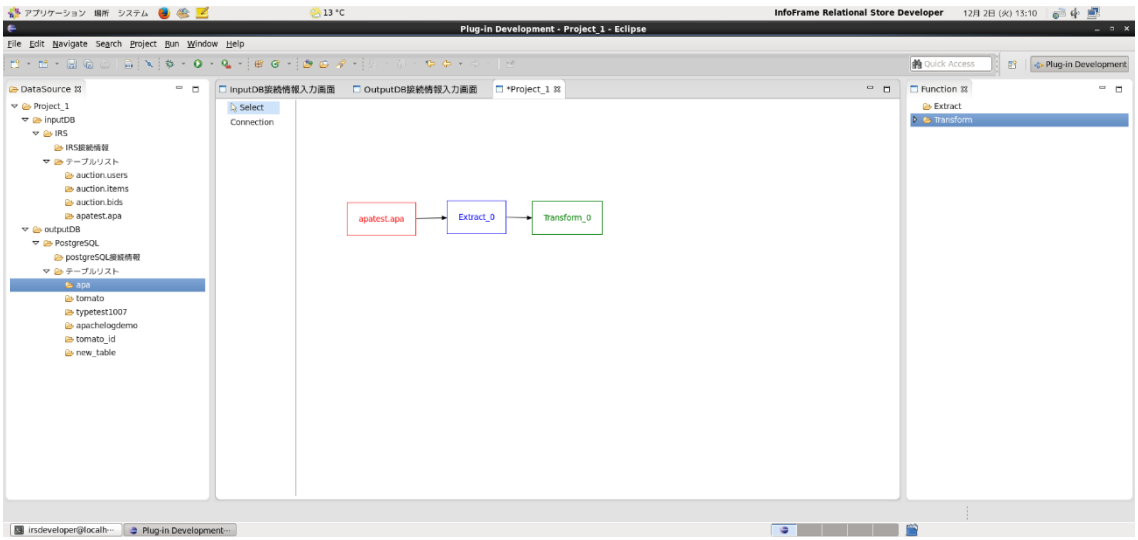
結果：削除したアイコンが復元される。



操作 6：作業ウィンドウを右クリックして、「Redo」を選択する。



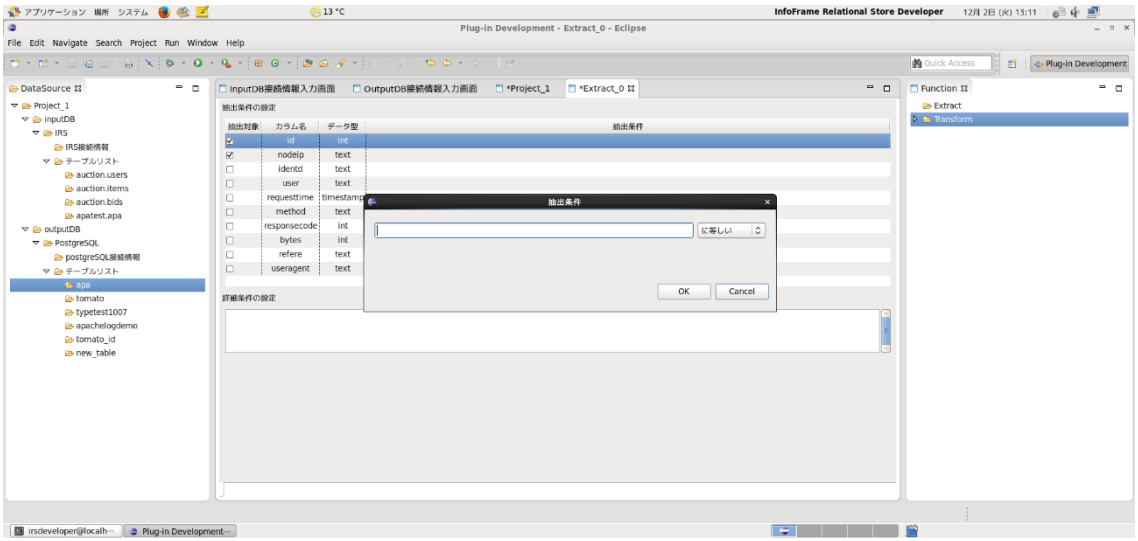
結果：復元されたアイコンはもう一度削除されている。



2.6. Extract 作業の編集

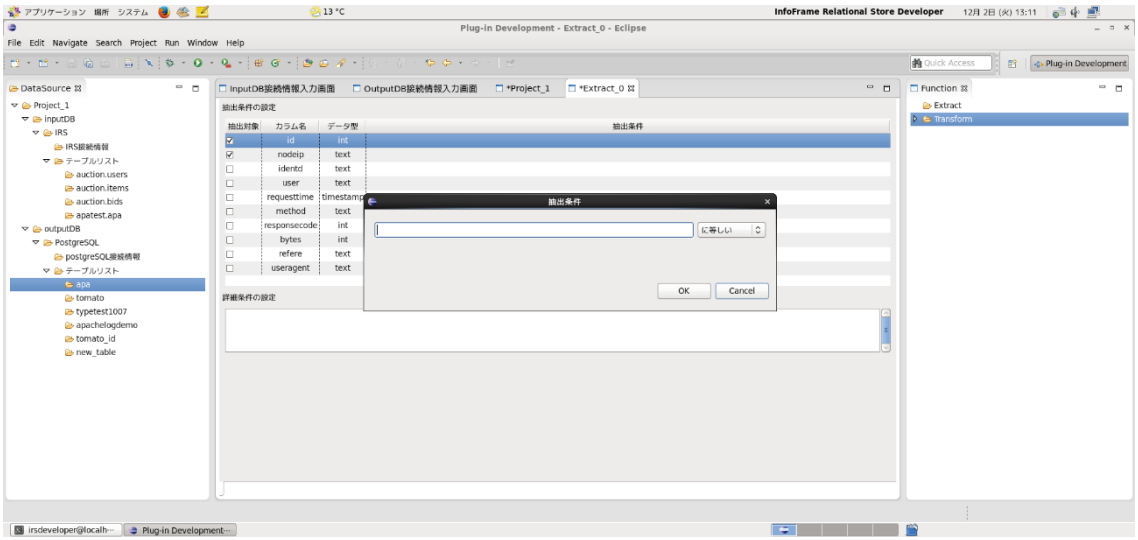
操作 1：Extract アイコンをダブルクリックする。

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

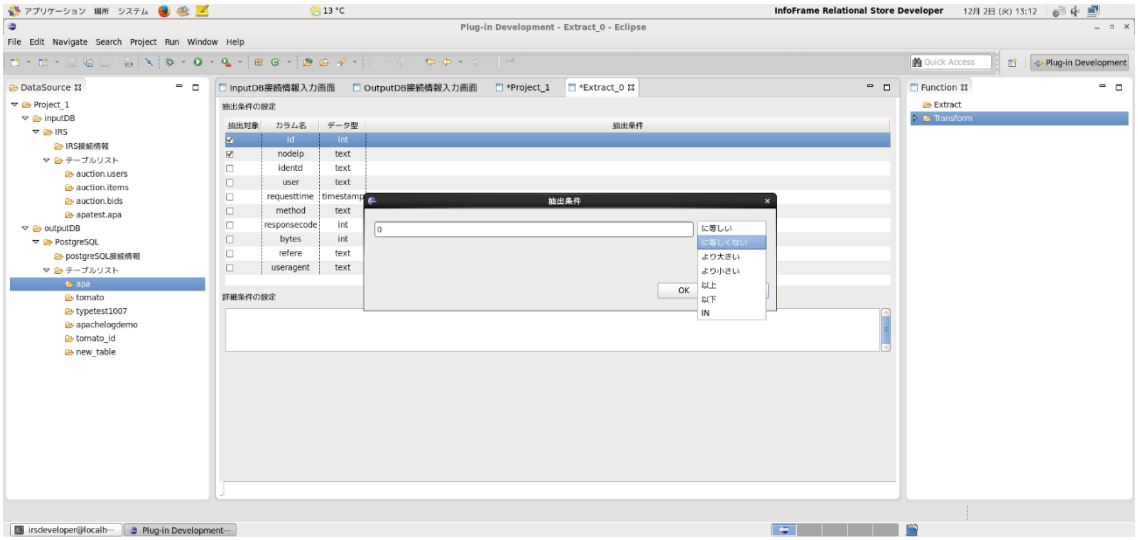


操作２：「抽出条件」のカラムをクリックします。

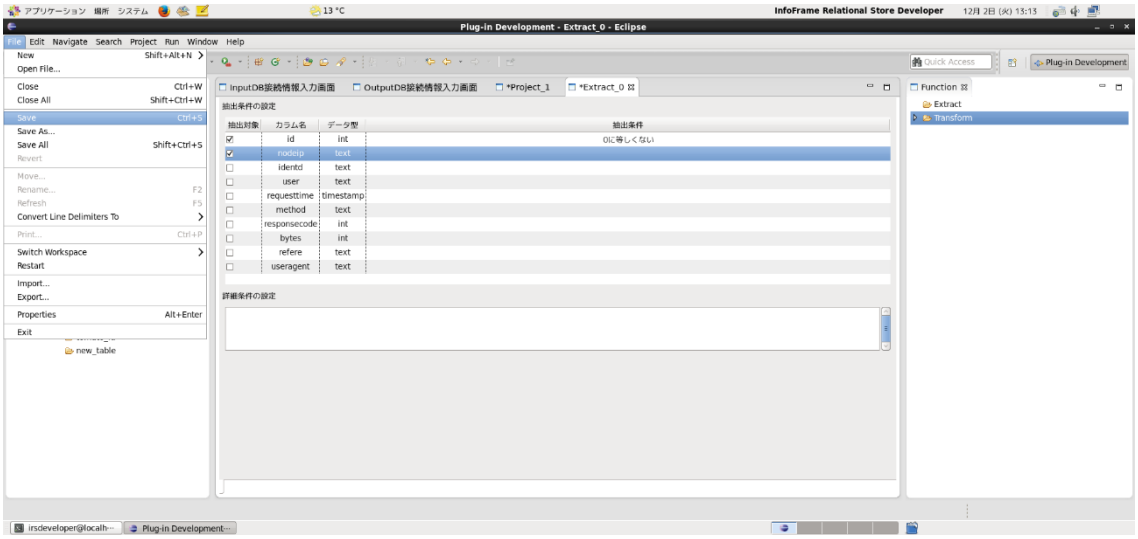
結果：抽出条件編集画面が表示される。



操作３：抽出条件を入力して、「OK」ボタンを押す。

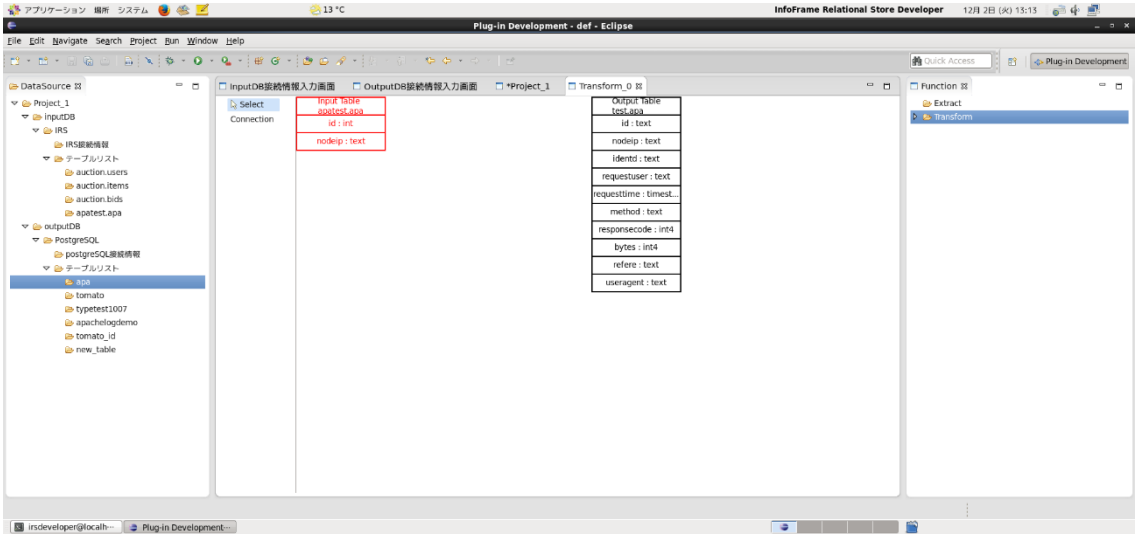


結果：Extract 編集画面に該当カラムの抽出条件が表示される。



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する。

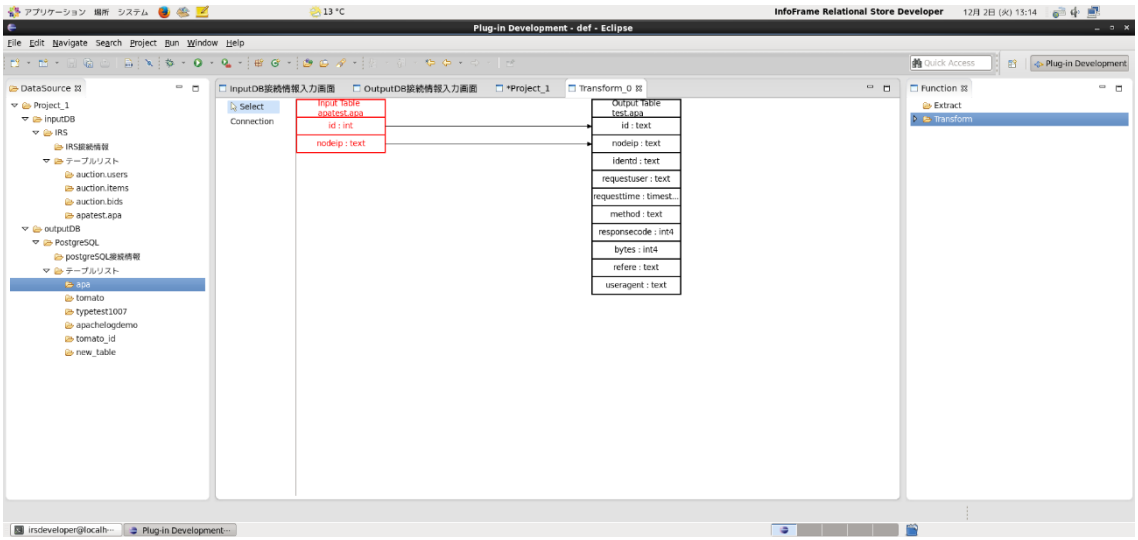
結果：Transform 編集画面に抽出後 Input Table が表示されている。



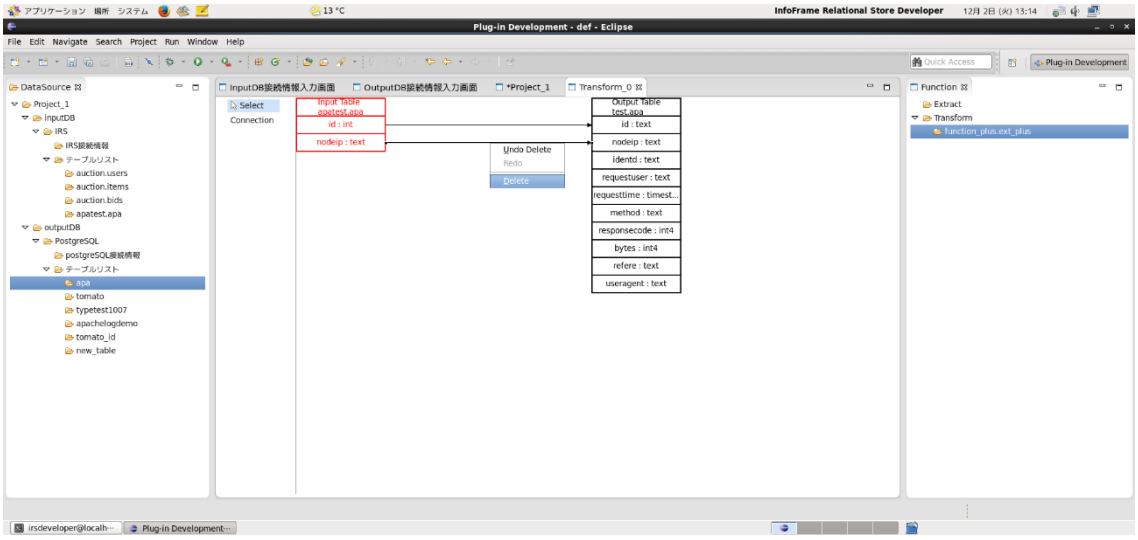
2.7. Transform 作業の編集

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する。

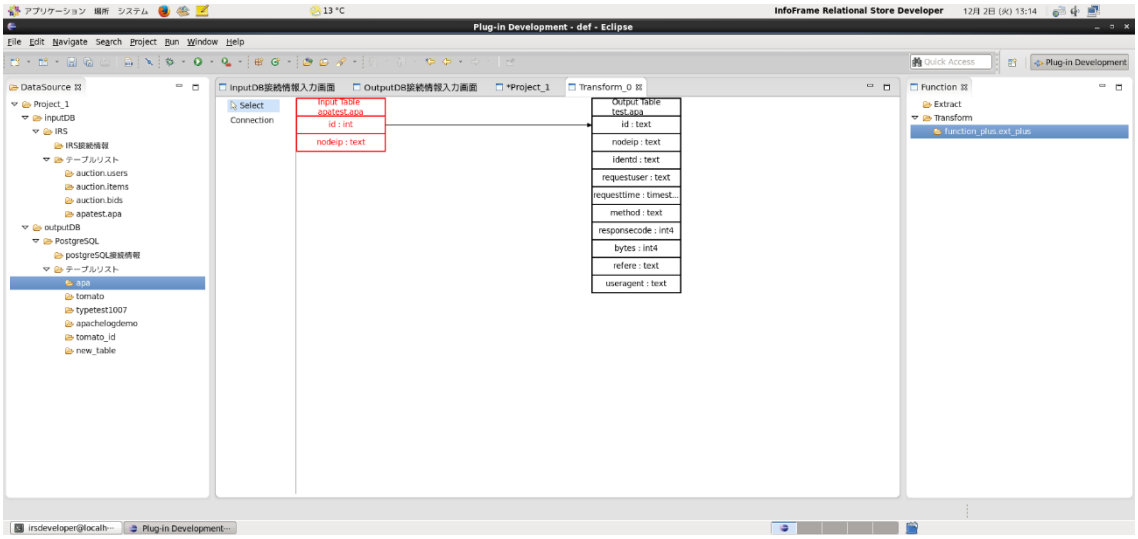
結果：対応関係を指定している線が表示されている。



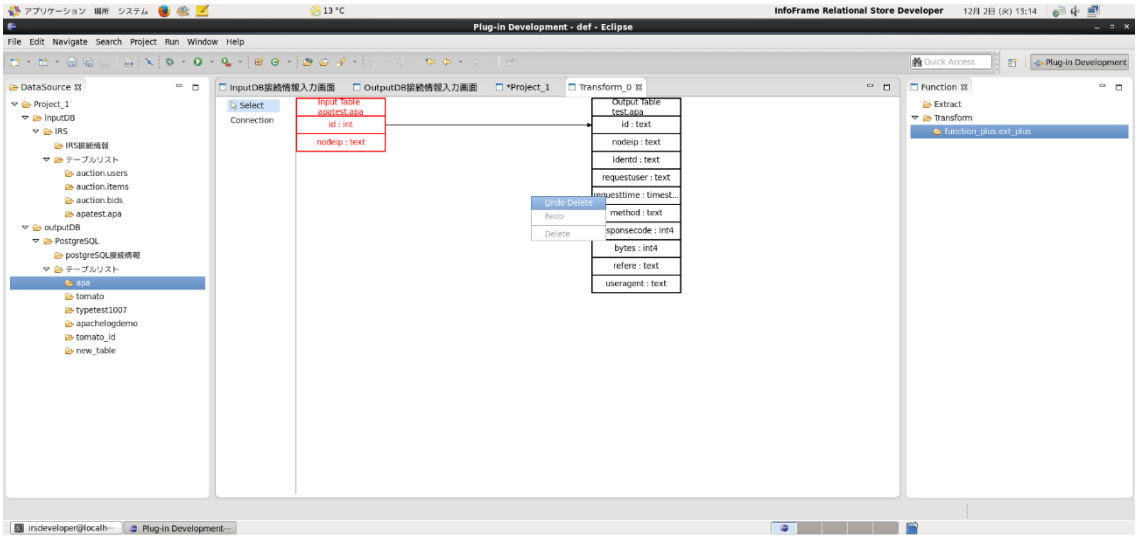
操作 2：線を右クリックして、「Delete」を選択する。



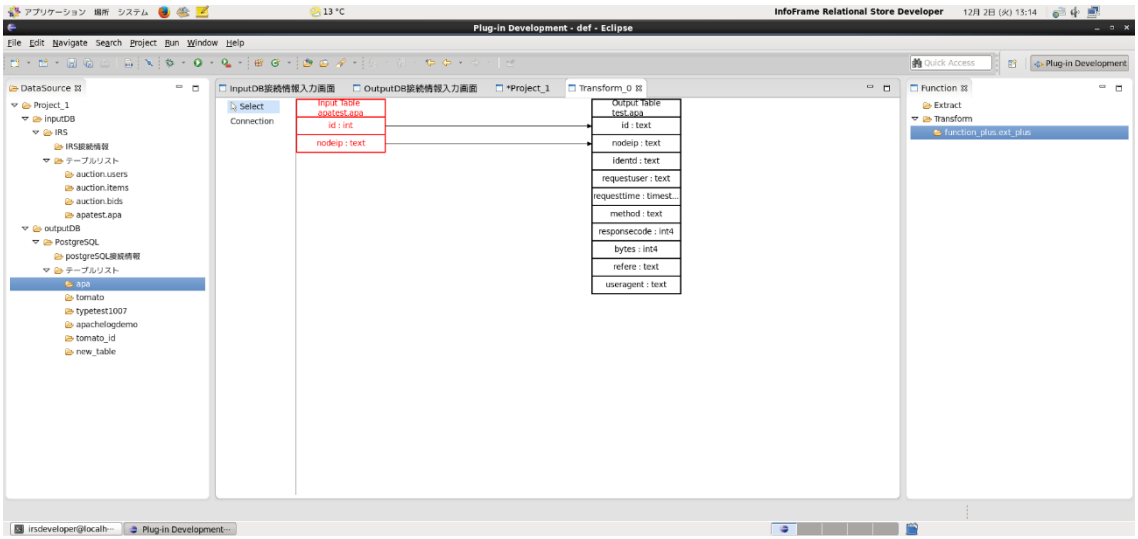
結果：選択されている線が消される。



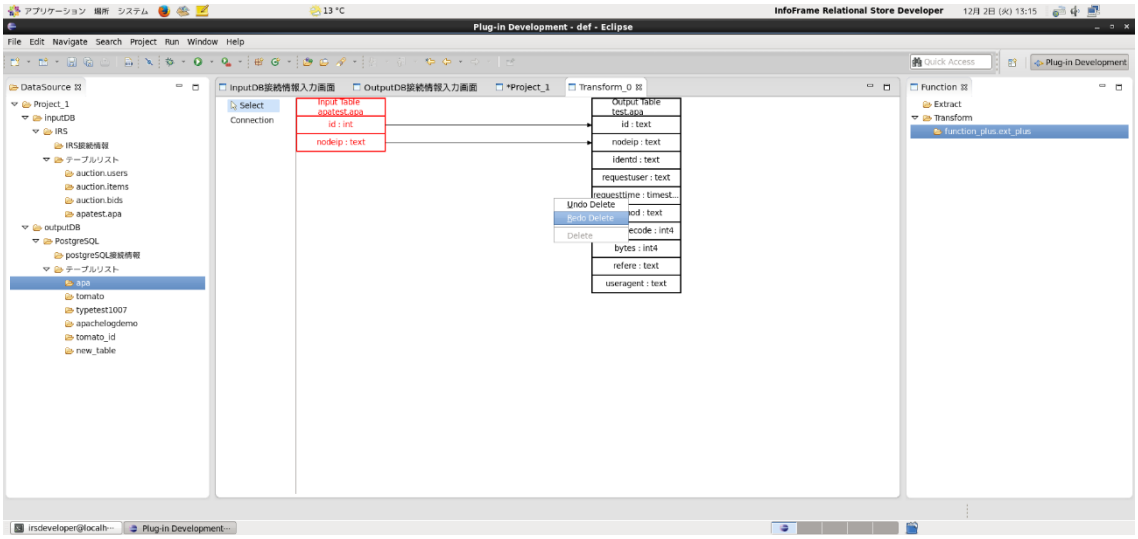
操作 3：Transform 編集画面を右クリックして、「Undo」を選択する。



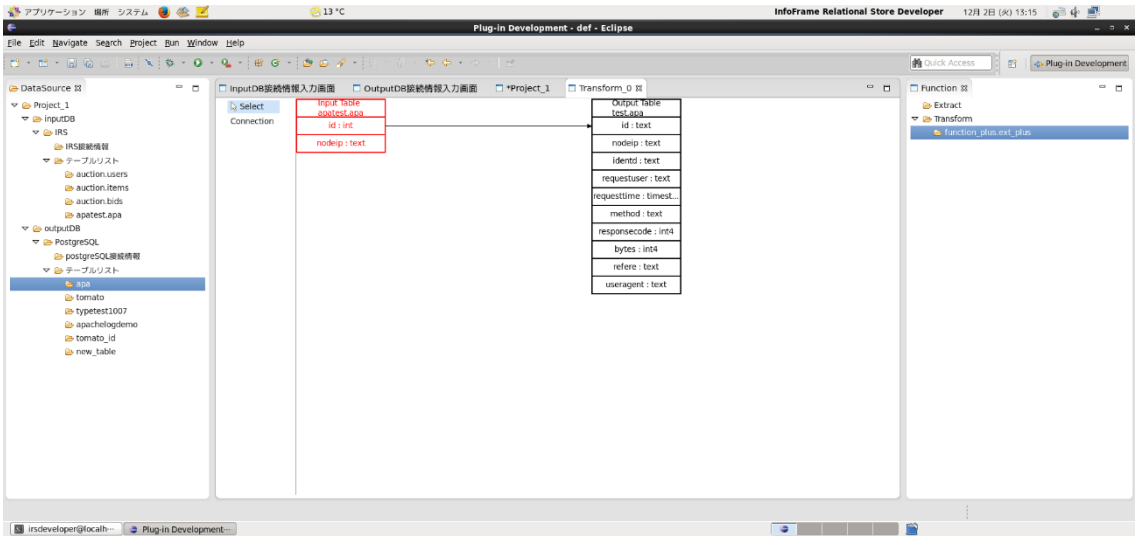
結果：削除された線が復元される。



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



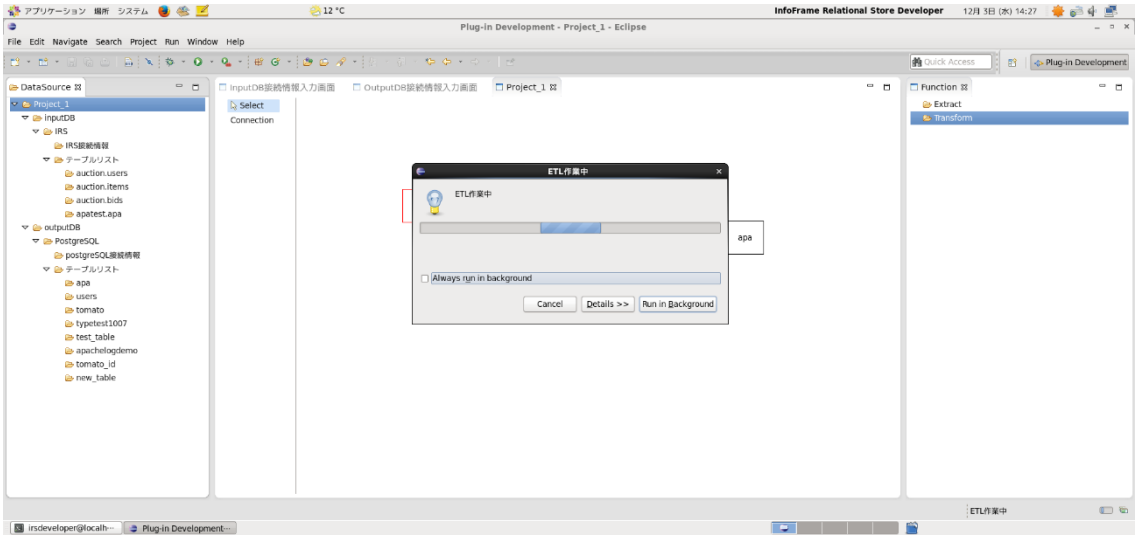
結果：復元された線がもう一度消される。



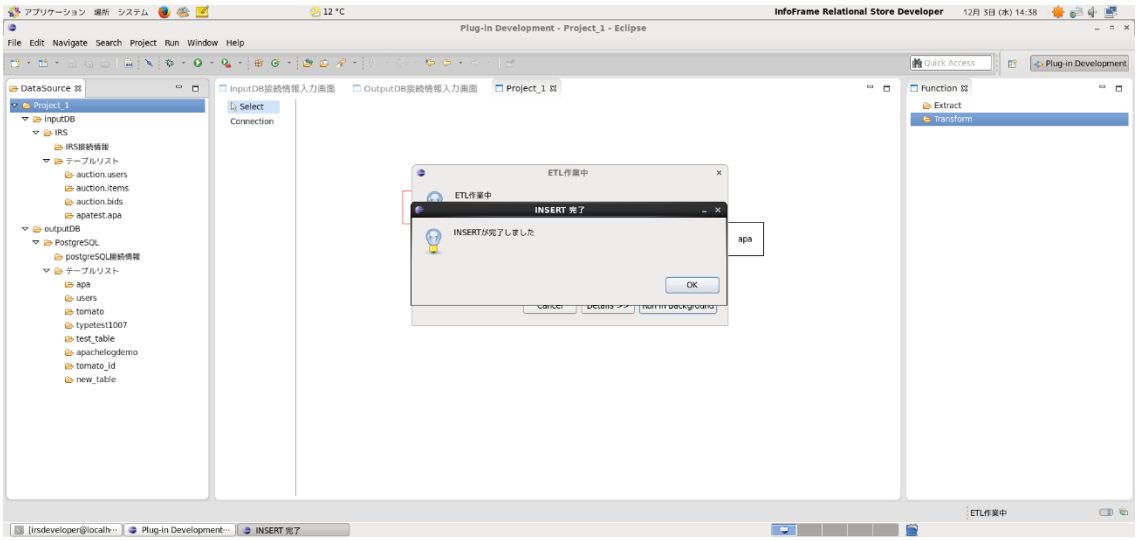
2.8. ETL の実行

操作：プロジェクト名を右クリックして、「実行」を選択する。

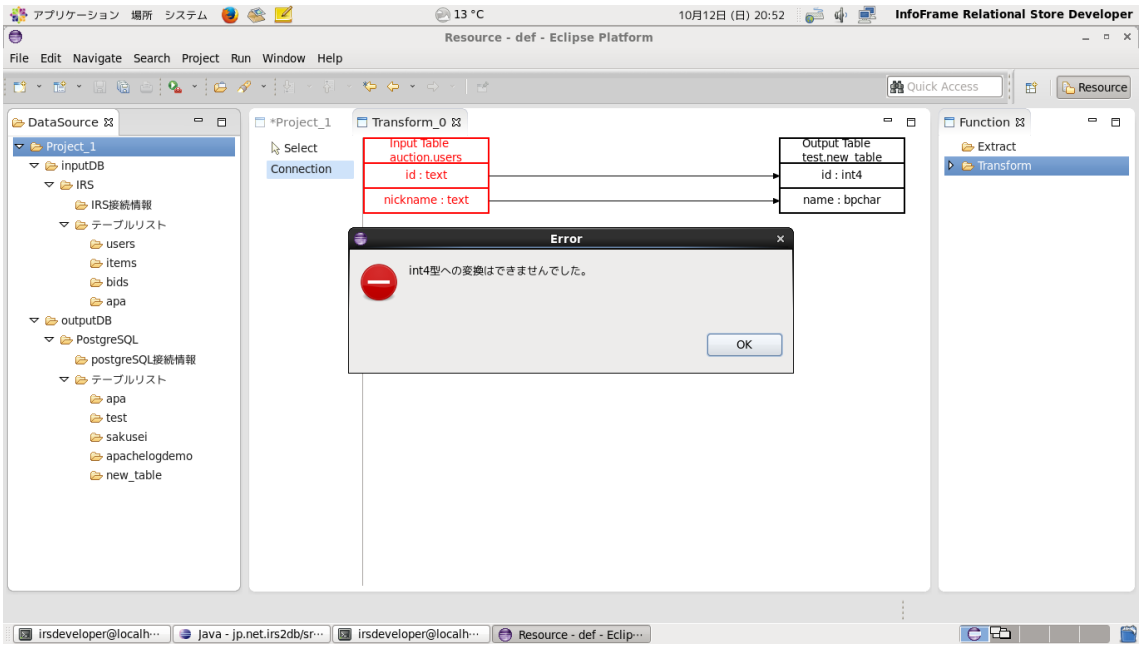
結果 A-1：ETL 実行中は進捗を通知するダイアログが表示される。



結果 A-2：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る。



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る。



以上のように、「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認した。

第６節　　第一イテレーション納品テストへの対応

第一イテレーションで顧客が行った受け入れテストの結果を以下に示す。

筑波大学研究開発プロジェクト第一イテレーション成果物
（10 月 15 日リリース）
受け入れテスト結果報告書

作成者：日本電気株式会社（並木）
作成日：2014 年 10 月 16 日

本文書では筑波大学研究開発プロジェクトの第一イテレーションの成果物に対し受け入れテストを実施した結果について報告する。

受け入れテストは要件定義（「納品報告書.docx」第 3 章に記載）に基づき、ユーザーズマニュアルから該当する操作方法を探して実施した。受け入れテストの結果は以下のとおり。

結果：要確認

以下に実施したテスト項目を列挙する。

結果に「*」を記載した項目は末尾に指摘事項を記載している。

==========
テスト項目
　　予想結果　　テスト結果
==========
1. IRS に接続する

1. IRS 接続情報をダブルクリックして表示される画面で適切な情報を入力し、「接続」ボタンをクリックすることで IRS に接続できること。
　　正しく接続できる　　○*1
----

2. IRS からテーブル定義データの取得

1. IRS に接続後、「テーブルリスト」に定義したテーブルが表示される。
テーブルリストに表示される ○*2

---

3. IRS から取得したテーブル定義データの表示

1. 主キーが表示される。
主キー列にチェックマークが表示される ×
2. カラム名が表示される
カラム名が表示される ○
3. データ型が正しく表示される
1. integer 型 ○
2. bigint 型 ○
3. float 型 ○
4. double 型 ○
5. decimal 型 ○
6. text 型 ○
7. char 型 ○
8. varchar 型 ○
9. boolean 型 ○
10. binary 型 ○
4. 長さが表示される
1. decimal 型 ×
2. char 型 ×
3. varchar 型 ×
5. NOT NULL 制約が表示される
表示される ×
6. サンプルデータが表示される
表示される ×

---

4. データを抽出するテーブルの指定

1. inputDB から作業ウィンドウにテーブルをドラッグ&ドロップし、Extract アイコ
ンと接続する。Extract アイコンをダブルクリックすると対象テーブルの情報が

表示される。

上記の操作が正しく行われる ○

---

5. IRS からデータを抽出する機能

12. データ挿入機能と合わせて評価する。

---

6. 抽出する際に抽出条件(フィルター)を指定する機能

本機能は第一イテレーションでは提供しないことにしたため、テスト項目はない。

---

7. PostgreSQL との接続

1. PostgreSQL 接続情報をダブルクリックして表示される画面で適切な情報を入力し、「接続」ボタンをクリックすることで PostgreSQL に接続できること。

正しく接続できる ○

---

8. PostgreSQL からテーブル定義データの取得テーブル

1. PostgreSQL に接続後、「テーブルリスト」に定義したテーブルが表示される。

テーブルリストに表示される ○

---

9. PostgreSQL から取得したテーブル定義データの表示

1. 主キーが表示される。

主キー列にチェックマークが表示される ○

2. カラム名が表示される

カラム名が表示される ○

3. データ型が正しく表示される

1. integer 型 ○

4. 長さが表示される

2. integer 型 ○

5. NOT NULL 制約が表示される

表示される ○

6. サンプルデータが表示される

表示される ○

7. 「テーブル作成」機能により作成したテーブルの定義データを表示する。

テーブル定義データが表示される ×*5

---

10. テーブル作成

1. 主キーの指定が反映される ○
2. カラム名の指定が反映される ○
3. データ型の指定が反映される
   2. char 型 ○
   3. varchar 型 ○
4. 長さの指定が反映される
   1. char 型 ○
   2. varchar 型 ×*4
5. NOT NULL 制約の指定が反映される ○
6. UNIQUE 制約の指定が反映される ○
7. すでに存在するテーブル名を指定する

テーブル作成時にエラーが発生する ○

8. 同じカラム名の列を複数定義する。

エラーが発生する ○

---

11. データを挿入するテーブルを指定する機能

1. outputDB から作業ウィンドウにテーブルをドラッグ&ドロップし、Transform アイコンと接続する。Transform アイコンをダブルクリックすると入力テーブルと出力テーブルの情報が表示される。

上記の操作が正しく行われる ○

---

12. データ挿入機能

1. プロジェクトを「実行」することにより、IRS の指定のテーブルの指定のカラムのデータが PostgreSQL の指定のテーブルにコピーされる。

上記の操作が正しく行われる ○

---

13. IRS から PostgreSQL 用にデータを変換する

1. IRS の double 型のカラムを PostgreSQL の int 型の列に対応付ける。

int 型に変換され、操作が正常終了する　　○*6

---

14. テーブル間の対応関係の定義を行う(マッピング)機能

1. 作業ウィンドウで Transform アイコンをダブルクリックし、以下の操作を行う。
   1. 入力列から出力列に向かって矢印を作成する　　○
   2. 出力列から入力列に向かう矢印は作成できない　　○
   3. 作成された矢印を削除する　　○
   4. 複数の入力列から単一の出力列に矢印を引く　　○
   5. 単一の入力列から複数の出力列に矢印を引く　　○

---

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

1. 作業ウィンドウに入力表をドラッグ&ドロップする　　○
2. 作業ウィンドウに出力表をドラッグ&ドロップする　　○
3. Function ビューからアイコンをドラッグ&ドロップする　　○
4. 作業ウィンドウの中でアイコンを移動する　　○

---

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

1. 入力表から Extract アイコンに線を引く　　○
2. Extract アイコンから Transform アイコンに線を引く　　○
3. Transform アイコンから出力表にアイコンを引く　　○
4. 入力表から Transform アイコンに線を引けない　　○
5. 入力表から出力表に線を引けない　　○
6. Extract アイコンから出力表に線を引けない　　○
7. 1 から 3 の逆向きに矢印を引けない　　○

==========

指摘事項

*1 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。

*2 データベースによる分類がないため、例えば db1 に t1、db2 に t1 と異なるデータベー
スに同名のテーブルを定義した場合、「テーブルリスト」には t1 が 2 つ並び、区別ができない点は、不親切である。

*3 「PostgreSQL 接続情報」の画面で PASSWORED とスペルミスがある。実用上の問題はな
し。

*4 データ型 varchar、長さ 3 としたカラムを定義し、「テーブルを作成」しても
PostgreSQL には長さ指定のない varchar 型のカラムが定義されているように見える。

*5 メニューから当該操作を選択すると、Eclipse の Error Log ビューに「Unhandled event loop exception」が記録される。

*6 ただし、型変換のルールがマニュアルに記載されておらず、仕様が不明。

評価環境

* CentOS 6.5 (x86_64)
* InfoFrame Relational Store V3.1
* PostgreSQL 8.4.20

以上

以上について第二イテレーションで対応した結果をいかに示す。

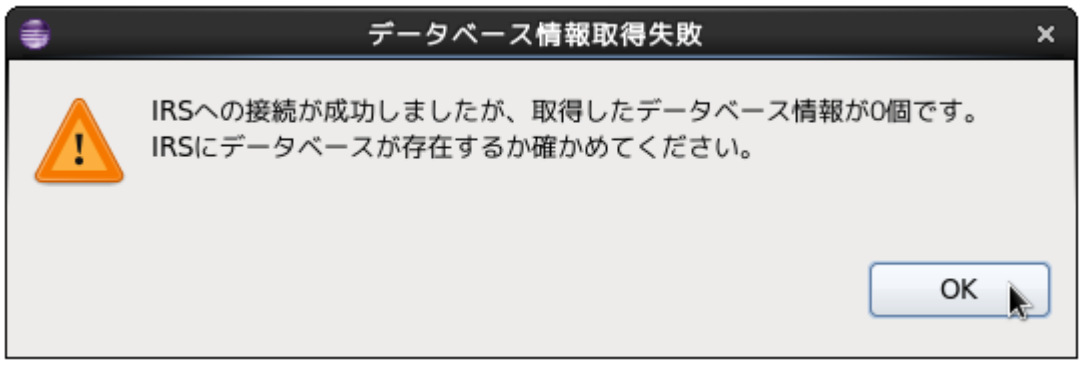
1. IRS に接続する
第 1 イテレーションでは IRS に接続する場合に表示されるエラー表示が一種類しかなく、問題があった。そのため、第 2 イテレーションでは IRS への接続時に 3 種類のエラー通知ダイアログを追加した。表示されるダイアログは以下の通りである。

① IRS への接続が失敗した

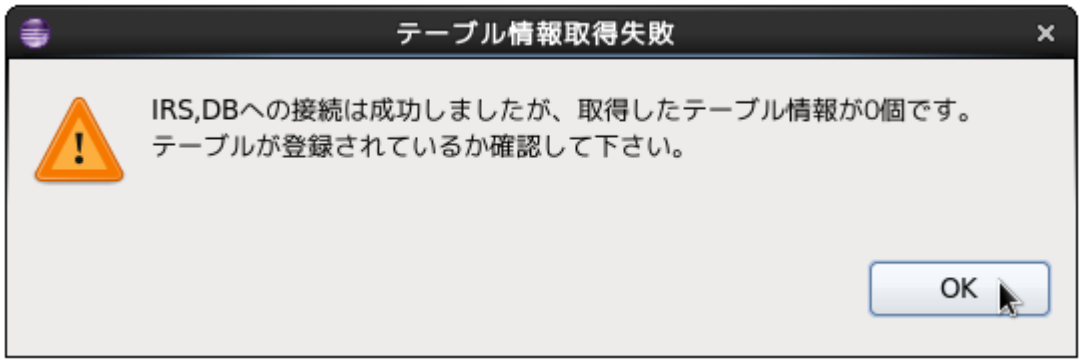
② (追加) IRS への接続は成功したが、取得した DB 情報が 0 個である
③ (追加) IRS、DB への接続は成功したが、取得したテーブル情報が 0 個である
④ IRS への接続が成功した
⑤ (追加) 既に IRS へ接続しており、再度取得する場合には再生成する必要がある

また、追加されたダイアログは以下画像の通りである。

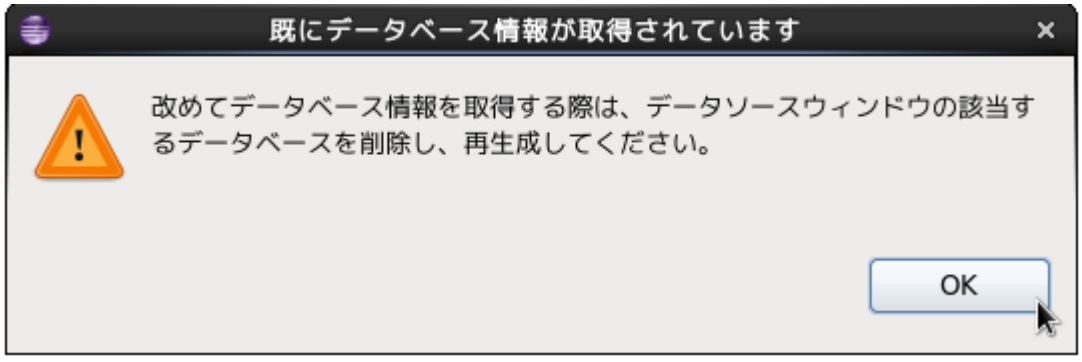
② IRS への接続が成功したが、取得した DB 情報が 0 個である



③IRS、DB への接続は成功したが、取得したテーブル情報が 0 個である



⑤既に IRS へ接続しており、再度取得する場合には再生成する必要がある



2. IRS から取得したテーブル定義データの表示

第１イテレーションでは IRS のテーブル定義データを表示する機能で一部要件定義とは違い、表示できないカラムがあった。これは調査不足によるもので、第２イレーションでは表示できないものは表示しない、もしくは空欄である。

表示するデータ

・カラム型

・データ型

表示しないデータ

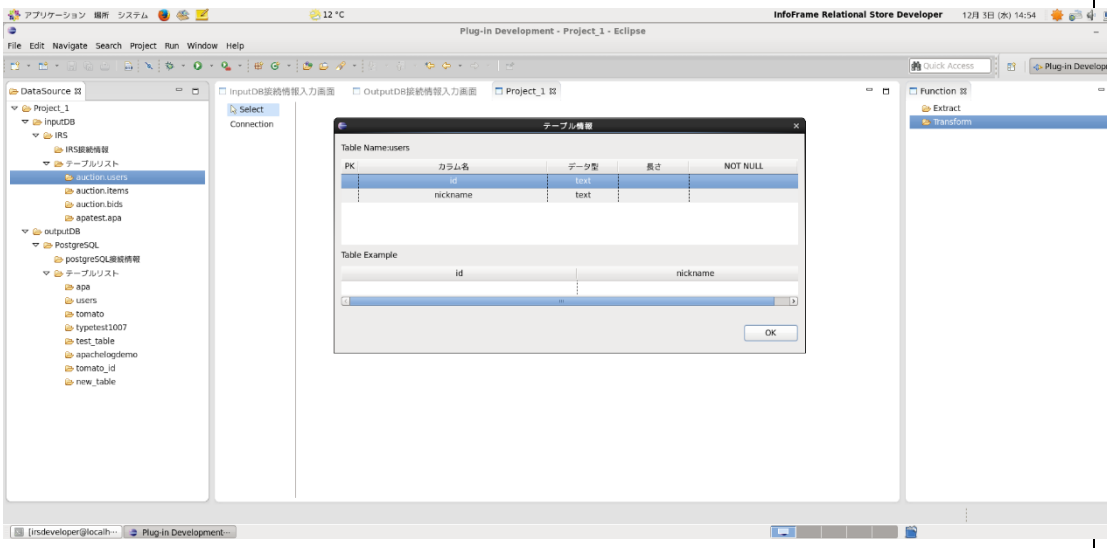
・PK

・長さ

・Not null

・Table Example

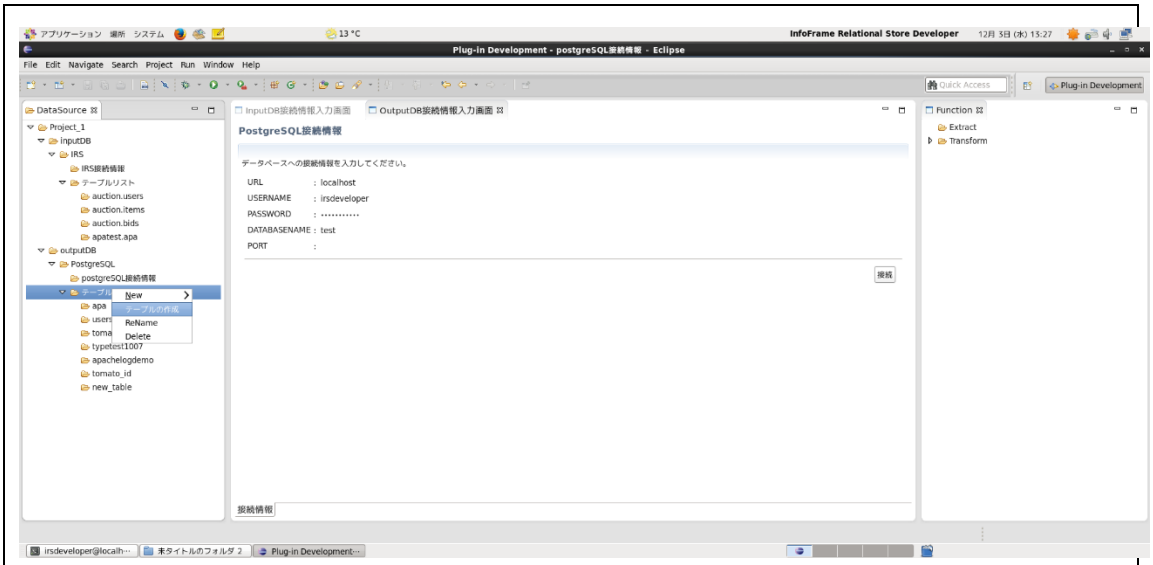
画面例は以下の通りである。



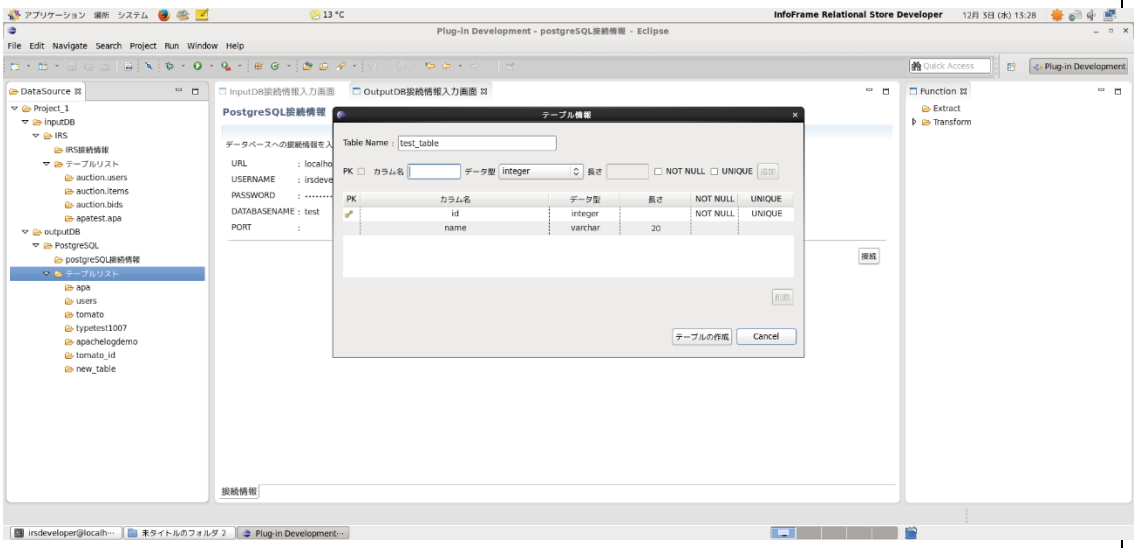
3.作成したテーブルの長さ指定について

第１イテレーションでは NET ツール上でテーブルを作成する際に長さの指定ができない型(varchar)が存在した。このバグを修正し、varchar 型の長さが指定できていることを以下に示す。

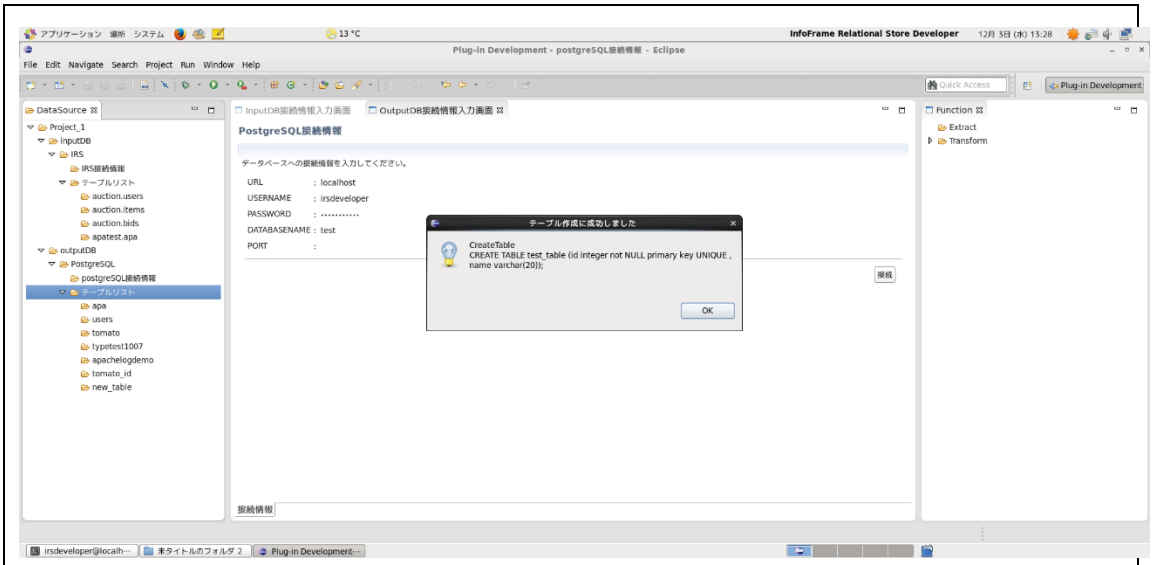
操作１:「テーブルリスト」を右クリックして、「テーブルの作成」を選択する。



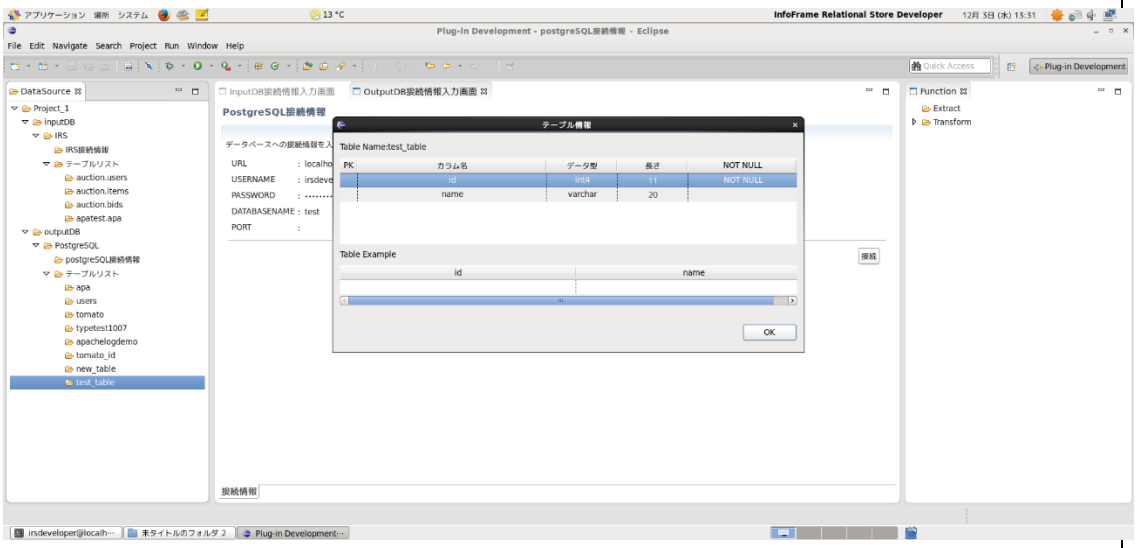
操作 2：「テーブル情報」で作成するテーブルの情報を入力して、「テーブ ルの 作成」ボタンを押す。



結果 A-1：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る。OK ボタンを押すと、テーブルリストにテーブル名が追加される。



結果 A-2：適切に作られていることを確認できる



その他

・IRS のテーブル名表示において「DB 名.テーブル名」と表示を変更した。

・データ型の変換一覧を作成した。

第 7 節　　第 1 イテレーション非機能要件調査報告書

本節では、設計段階で合意した非機能要件について実現されているかの調査を報

告する。

第１項　非機能要件達成度一覧

性能：「ロード性能は、PostgreSQL にシングルスレッドでインサートするよりも早いこと」達成されている。

「ツールを使用したデータ移行にかかる時間は、IRS からファイルを経由してデータ移行する場合より短いこと」達成されている。

運用：ユーザマニュアルは作成されている。

保守性：セットアップマニュアル、javadoc は作成されている。

移行性：CentOS 依存の仕様は

セキュリティ：ユーザが入力したパスワードは画面に表示されない。今回、プロジェクトの保存機能は実現していないため、入力されたパスワードなどへの暗号化は実施していない。

第２項　ロード性能調査詳細報告

本項では、「ロード性能は、PostgreSQL にシングルスレッドでインサートするよりも早いこと」に対する調査の詳細を報告する。

- 実験に使用したデータ

Apache のログデータ(combined 形式)の形式を模したデータである。表１にカラム名とカラム型、説明を示す。実験に使用するテーブルは実験一回ごとに削除・作成する。

表 4

カラム名	カラム型	説明
ID	integer	主キーとして扱うカラム
nodeIP	text	アクセス要求元の IP アドレス
identd	text	リモートログ名（基本"-"）
user	text	リモートユーザ名 (Basic 認証のユーザ)
requesttime	timestamp	リクエスト受付時刻
method	text	リクエストの最初の行
responsecode	integer	レスポンスコード
bytes	integer	レスポンスバイト数
refere	text	リファラー
useragent	text	ユーザエージェント

表２に実験に使用したデータを示す

表 5

ID	1
nodeIP	130.158.81.75
identd	-
user	-
requesttime	[02/Jul/2013:13:33:37 +0900]
method	"GET /redmine/ HTTP/1.1"
responsecode	200
bytes	3026
refere	"http://vm05.sit.cs.tsukuba.ac.jp/redmine/"
useragent	"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36"

このデータの ID のみ変動させ 100 万行のデータを用意する。その後以下の手法を用いて INSERT し、時間を計測する。

1.通常の INSERT 文を作成し、INSERT する。

2.PreparedStatement を使用する。

- 実験環境

実行マシン：高度 IT 支給のノート PC

OS：VMWare 上の CentOS6.5 64bit

CPU：Intel(R) Core(TM) i5-3320M 2.60GHz

メモリ：1.8G バイト

データベース：PostgreSQL 8.4.20 on x86_64-redhat-linux-gnu, compiled by GCC gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-4), 64-bit（実行マシン上で動作している）

JDBC:：Version 8.4 Build 703

- 実験の流れ

図 1 に、実験の流れを説明する。

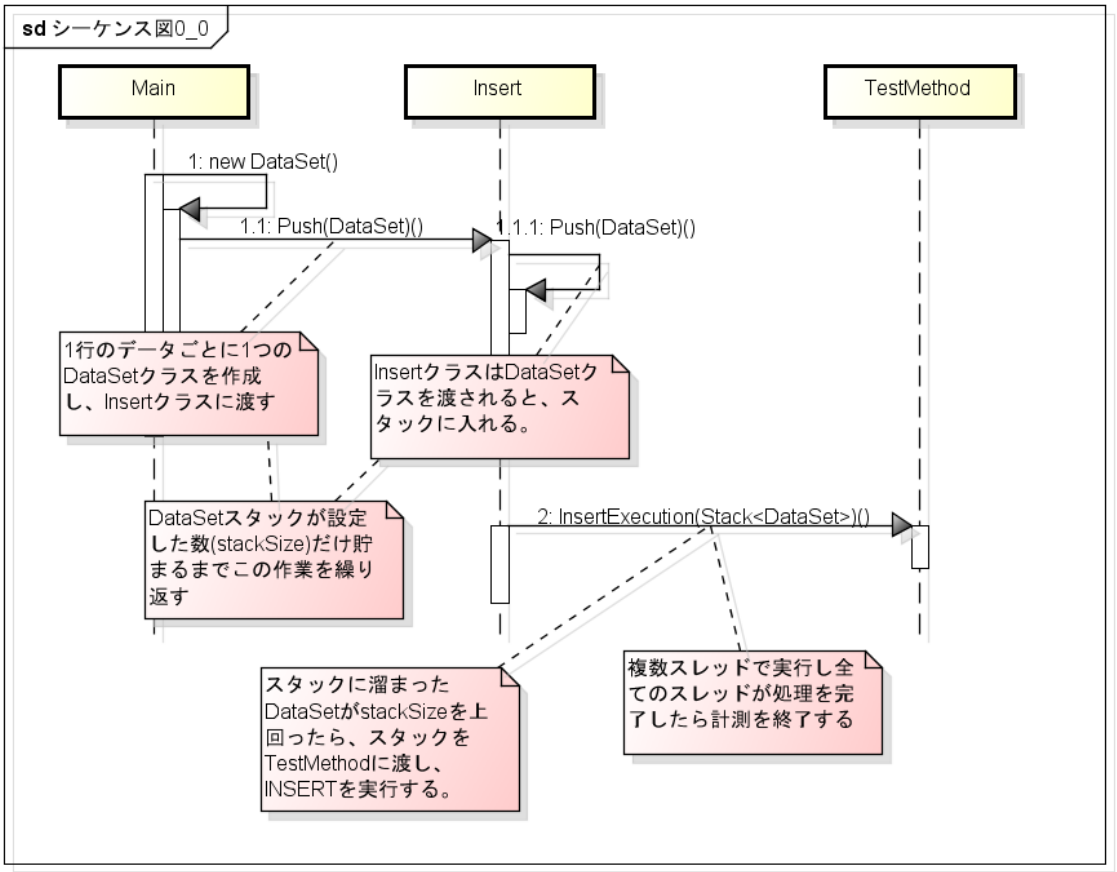


図 31

Main クラス：指定した数(100 万行)の INSERT データを作成する。本実験では 1 行のデータに付き、1 つの DataSet クラスを作成し使用する。

Insert クラス：スレッド管理を行う。また、Main から渡された DataSet をスタックに溜め、指定した量が溜まった時点で TestCrass クラスのインスタンスを作成し DataSet スタックを渡す。その後新しい DataSet スタックを作成する。

TestCrass：JDBC を用いて渡された DataSet スタックのデータの INSERT を行う。通常の INSERT 文を用いるか PreparedStatement を用いるかをこのクラスを変更することで選択する。本クラスはマルチスレッドで実行される。

計測範囲：Main クラスが一つ目の INSERT データを作成し始めた時点から、全てのスレッドが終了するまでの時間を計測する。

- JDBC のコミット

JDBC は、デフォルトでは全ての SQL 文は、個別のトランザクションとしてコミットされるようになっている(自動コミットモード)。今回は、1.高速な処理が求められること、2.ETL の機能上、データ移行作業中のデータベースから読み出し作業は行われないこと、から自動コミットモードを OFF にし、渡された DataSet ス

タック全ての INSERT を終えてからコミットすることにした。

- JDBC バッチ処理機能

高速化を目的とし、JDBC のバッチ処理機能(Statement の addBatch と executeBatch)を使用した。複数の SQL 命令をひとまとめにして実行する機能である。今回は、30 行の INSERT を一つのバッチとして処理する。

- 調査対象の設定項目

1. 最大スレッド数

TestCrass が実行されるスレッド数の最大数を 1,2,4,6,8,10 と変動させそれぞれで時間を調査した。スレッド管理には Java の
java.util.concurrent.Executors.newFixedThreadPool を使用した。
newFixedThreadPool は最大スレッド数を指定し、その範囲内で新規スレッドを作成する。スレッド数が最大スレッド数に達した場合に、追加のタスクが送信されると、それらのタスクはスレッドが使用可能に成るまで、キューで待機する。

2. INSERT データの分割数

100 万行ある INSERT データを複数スレッドで実行する場合、データを分割する必要がある。そこで、分割する大きさによって INSERT にかかる速度が変化するのか確かめる。分割数は、1,10,20,50,100 とした。1 の場合は 100 万行データを 1 つのスタックに溜める。そのスタックを TestMethod に渡し、スタックから取り出しながら INSERT を行う。10 の場合は 100 万行のデータを容量 10 万のスタックに保存する。スタックが一杯になるとそれらを複数スレッドで実行している TestMethod に渡し、スタックから取り出しながら INSERT を行う。分割数が 1 の場合は Testmethod が実行されるスレッド数は 1 のため、その他にスレッド数を増やした実験は行わなかった。

以下に計測した結果をグラフにしたものを表示する。
図 2 ではそれぞれ三回計測し、それの平均をグラフ化したものである。
1,2,4,6,8,10 はスレッド数を示している。
1,10,20,50,100,P1,P10,P20,P50,P100 はデータ(1000000 行)を何分割しているかを示している。先頭に P が付いているものは PreparedStatement を使用した結果である。

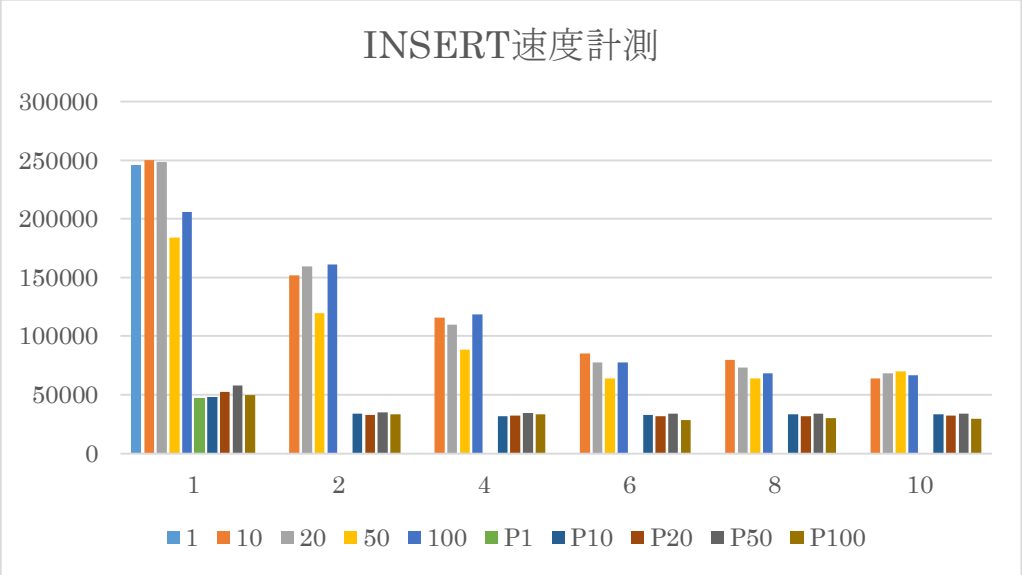
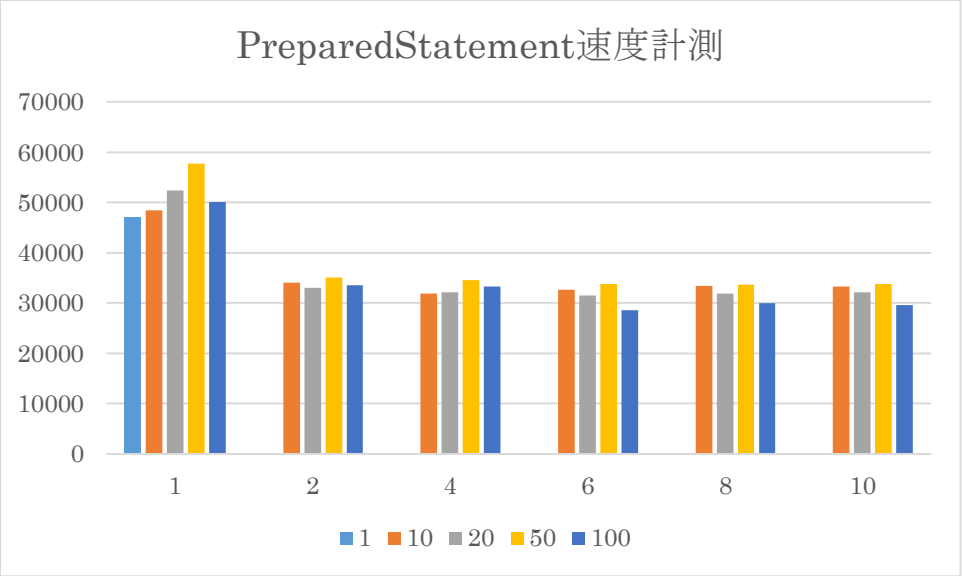


図 32

図 3 は図 2 から PreparedStatement の結果のみを抜き出したグラフである。



以下に計測した結果のまとめを記載する。

- 通常 INSERT
  - スレッド数を増やすと高速になる
  - 分割数を増やすとやや高速になる。分割数を増やしすぎると低速になる。
  - ロード性能についての非機能要件は、通常 INSERT の手法でスレッド数を増やすことで達成できた。
- Preparedstatement
  - スレッド数を増やしても速度はあまり変わらない

- 分割数を増やした場合、1 スレッドと複数スレッドでは複数スレッドの方が高速だったが、さらにスレッド数を増やしてもあまり変わらない
- 今回の実験ではどの設定においても通常 INSERT よりも高速だった。

- 総評

  今回の実験では、PreparedStatement を使用した場合、通常の INSERT よりも高速だった。PreparedStatement を使用することで、顧客と合意した非機能要件である「ロード性能は、PostgreSQL にシングルスレッドでインサートするよりも早いこと」という目標を達成できると判断する。

第３項　ツール使用時間に関する調査

本節では「ツールを使用したデータ移行にかかる時間は、IRS からファイルを経由してデータ移行する場合より短いこと」に対する調査の詳細を報告する。

- 実験に使用するデータ：Apache のログデータ(combined 形式)の形式を模したデータ(10 万行)

データ移行先のテーブルは実験一回ごとにテーブル削除と作成を行う。

カラム名	カラム型	説明
ID	integer	主キーとして扱うカラム
nodeIP	text	アクセス要求元の IP アドレス
identd	text	リモートログ名 (基本"-")
user	text	リモートユーザ名 (Basic 認証のユーザ)
requesttime	timestamp	リクエスト受付時刻
method	text	リクエストの最初の行
responsecode	integer	レスポンスコード
bytes	integer	レスポンスバイト数
refere	text	リファラー
useragent	text	ユーザエージェント

1.1NET ツールを使用しない場合。

IRS に JDBC を用いて接続し、テスト用テーブルからデータを入手する。入手したデータを一時ファイルに作成しする。その後、psql ツールを用いて PostgreSQL に INSERT する。

①　IRS からの抽出：

作成したプログラムを使用し IRS からテスト用データを抽出する。抽出したデータを CSV ファイルに出力する。

② CSV ファイルの移動：

ptql で読み込める場所にに CSV ファイルを移動する

③ CSV ファイルの読み込み：

ptql ツールで csv ファイルを読み込む

(COPY apa FROM '/var/lib/pgsql/data/result.txt' WITH CSV;)

結果

作業	操作完了時の 0.スタートからの経過時間(分:秒)
0.スタート	0:00
1.IRS からの抽出	3:16
2.CSV ファイルの移動	4:30
3.CSV ファイルの読み込み	4:56

(経過時間にはコマンドを打ち込む時間などを含む)

1.2NET ツールを使用する場合

NET ツールを使用し IRS から PostgreSQL に対象データを移行する。

操作内容

① プロジェクトの作成：
プロジェクトを作成する

② テーブルの準備：
データ移行元の IRS とデータ移行先の PostgreSQL に接続し、データベース、テーブル情報を取得する。

③ 抽出対象の設定：
全てのカラムを抽出対象に設定する。

④ カラム間の対応関係の設定：
データ移行元テーブルとデータ移行先テーブルのカラムの対応を設定する。

⑤ データ移行の開始：
データ移行を開始する

⑥ データ移行完了の確認:
データ移行が完了したことを確認する

作業	操作完了時の 0.スタートからの経過時間

	(分:秒)
0.スタート	0:00
1.プロジェクトの作成	0:04
2.テーブルの準備	0:55
3.抽出対象の設定	1:12
4. カラム間の対応関係の設定	1:30
5. データ移行の開始	1:36
6. データ移行完了の確認	4:54

(経過時間にはコマンドを打ち込む時間などを含む)

総評：

ツールを使用した場合の方が、Java でプログラミングをしてデータを移行した場合よりもデータ移行にかかる時間は短かった。

Java でのプログラムングでデータ移行を行う場合は、Java でのプログラムや JDBC, IRS の仕様, ptql に関する知識が必要とされ、調査も含めてプログラミングには３０分ほどの時間が必要だった。一方、ツールを使った場合は、ツールに対する知識のみでデータ移行を行うことができる。よって、ツールを使用したほうがユーザの負担が少ないと考えられる。

以上より、「ツールを使用したデータ移行にかかる時間は、IRS からファイルを経由してデータ移行する場合より短いこと」という非機能要件の目標について、データ移行にかかる時間は同等かやや短い、との実験結果より、達成されていると考えられる。また、ユーザの負担を本ツールが軽減していることも実験の結果示されたと考えられる。

第８節　　第２イテレーション非機能要件達成度調査

第１項　ロード性能調査詳細報告

本項では、「ロード性能は、PostgreSQL にシングルスレッドでインサートするよりも早いこと」に対する調査の詳細を報告する。

- 実験に使用したデータ

  第 7 章第 6 節第 2 項と同様のため省略する。

- 実験を行った手法

  1.PreparedStatement を使用する。(第一イテレーションで実装した方法)

  2.COPY を使用する。

- 実験環境

  第 7 章第 6 節第 2 項と同様のため省略する。

- 実験の流れ

  第 7 章第 6 節第 2 項と同様のため省略する。

- JDBC のコミット

  第 7 章第 6 節第 2 項と同様のため省略する。

- JDBC バッチ処理機能

  高速化を目的とし、JDBC のバッチ処理機能(Statement の addBatch と executeBatch)を使用した。複数の SQL 命令をひとまとめにして実行する機能である。今回は、30 行の INSERT を一つのバッチとして処理する。バッチ処理は PreparedStatement のみ使用した。

- 調査対象の設定項目

  第 7 章第 6 節第 2 項と同様のため省略する。

- 計測結果

以下に計測した結果をグラフにしたものを表示する。
図 2 ではそれぞれ三回計測し、それの平均をグラフ化したものである。
1, 2, 4, 6, 8, 10 はスレッド数を示している。
C1, C10, C20, C50, C100, P1, P10, P20, P50, P100 はデータ(1000000 行)を何分割しているかを示している。先頭に P が付いているものは PreparedStatement を使用した結果である。先頭に C が付いているものは COPY を使用した結果である。

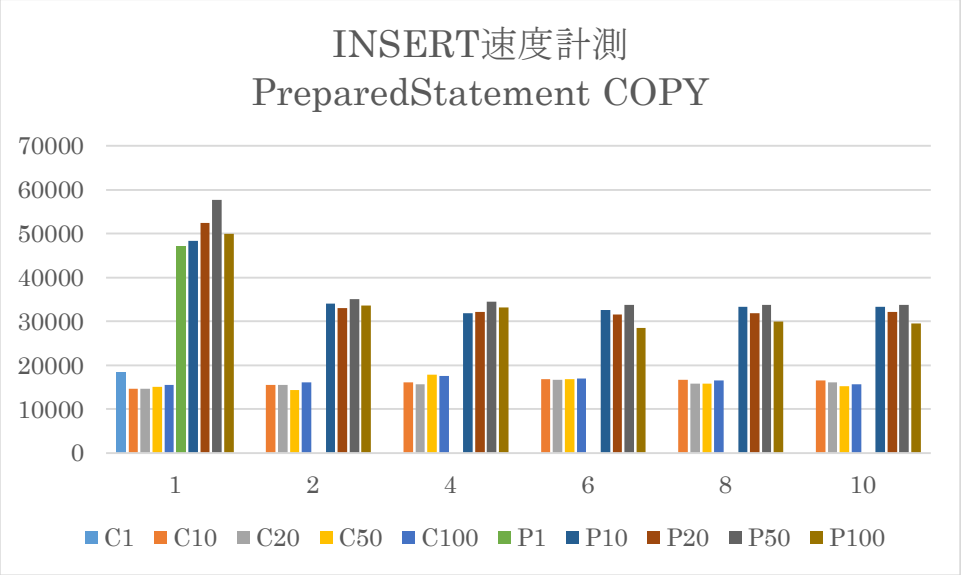
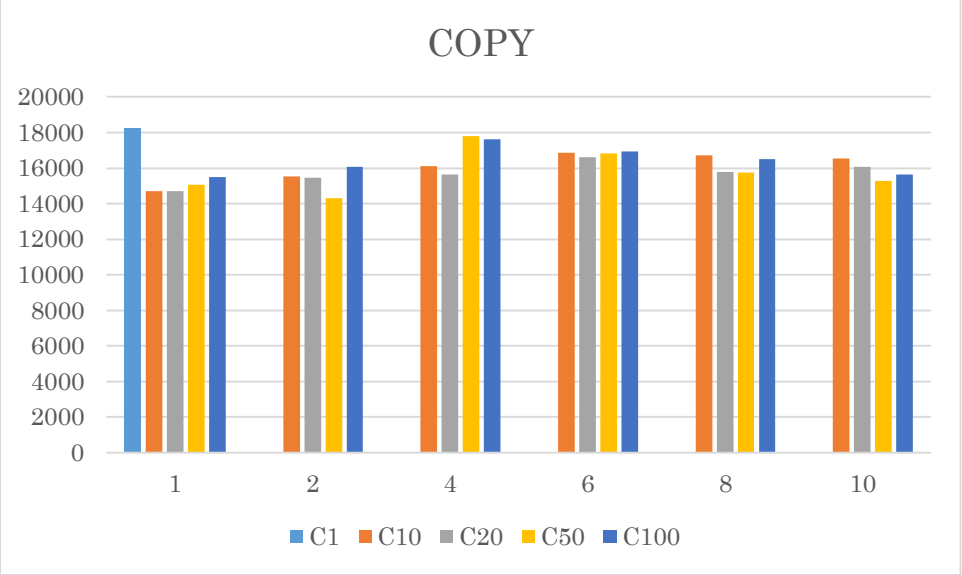


図 33

図 3 は図 2 から COPY の結果のみを抜き出したグラフである。



以下に計測した結果のまとめを記載する。

- PreparedStatement
  - スレッド数を増やしても速度はあまり変わらない
  - 分割数を増やした場合、1 スレッドと複数スレッドでは複数スレッドの方が高速だったが、さらにスレッド数を増やしてもあまり変わらない
- COPY
  - スレッド数, データ分割数を変化させてもデータの挿入速度はあまり変化がなかった。
  - 今回の実験ではどの設定においても PreparedStatement よりも高速だった。

- 総評

  今回の実験では、第一イテレーションで採用した PreparedStatement よりも COPY を用いた実装の方が高速であることが判明した。第二イテレーションの実装では COPY を用いることにした。

第２項　ツール使用時間に関する調査

本節では「ツールを使用したデータ移行にかかる時間は、IRS からファイルを経由してデータ移行する場合より短いこと」に対する調査の詳細を報告する。第２イテレーションでは第１イテレーションと使用したデータの量や時間計測のタイミングなどが異なる。

- 実験に使用するデータ：Apache のログデータ(combined 形式)の形式を模したデータ(100 万行)

データ移行先のテーブルは実験一回ごとにテーブル削除と作成を行う。

カラム名	カラム型	説明
ID	integer	主キーとして扱うカラム
nodeIP	text	アクセス要求元の IP アドレス
identd	text	リモートログ名 (基本"-")
user	text	リモートユーザ名 (Basic 認証のユーザ)
requesttime	timestamp	リクエスト受付時刻
method	text	リクエストの最初の行
responsecode	integer	レスポンスコード
bytes	integer	レスポンスバイト数
refere	text	リファラー
useragent	text	ユーザエージェント

1.NET ツールを使用しない場合

1.1NET ツールを使用しない場合の概要

事前に、以下の二つの機能を持ったプログラムを作成する。

1.IRS に JDBC を用いて接続し、データを抽出する。

2.抽出したデータを CSV 形式でファイルに書き込む。

作成したプログラムを実行し、CSV ファイルを作成する。

その後､psql ツールを用いて作成した CSV ファイル中のデータを PostgreSQL に INSERT する。

1.2 操作内容

④ IRS からの抽出：

作成したプログラムを使用し IRS からテスト用データを抽出する。抽出したデータを CSV ファイルに出力する。

⑤ CSV ファイルの移動：

ptql で読み込める場所に CSV ファイルを移動する

⑥ CSV ファイルの読み込み：

ptql ツールで COPY コマンドを用いて CSV ファイルを読み込む

1.3 結果

作業	かかった時間(分：秒)
1. IRS からの抽出	32 分 47 秒
2. CSV ファイルの移動	43 秒
3. CSV ファイルの読み込み	30 秒
合計	34 分

(経過時間にはコマンドを打ち込む時間などを含む)

1.4 NET ツールを使用しない場合まとめ

IRS からのデータの抽出に時間がかかった。今回は、IRS のマニュアルの 1 つである「sql-reference.pdf　5.5.1JDBC サンプルプログラム」を参考にし、PreparedStatement を用いたデータの抽出を行った。

このプログラムの作成は実験担当が行い、20 分程度の時間がかかった。実験担当は NET ツールの Load 機能の実装を行った経験から、JDBC や IRS の仕様にある程度の理解があるためこの程度の時間でプログラミングを行えたと考える。JDBC や IRS の仕様について習熟していない人がプログラムを作成する場合はこれ以上に時間がかかると考えられる。

NET ツールを使用しない場合は、ETL 作業にかかる時間の内大部分をデータの抽出が占めている。データの抽出と CSV ファイルの作成の後はほとんど時間がかかっておらず、データの挿入は PostgreSQL の COPY コマンドで 30 秒と高速なものだった。

CSV ファイルのファイルサイズは今回約 80 メガバイトのサイズがあり、ディレクトリ間のコピーなどにやや時間がかかった。100 万行のデータはビッグデータとしてはごく小規模なものであり、一般的なビッグデータを CSV ファイルにする場合はファイルの容量に注意しなければなない。

抽出元データベースと挿入先データベースの情報やカラムの情報を調べるために、IRS の ptql、PostgreSQL の psql を用いる必要があり、情報の収集が容易ではなかった。

2. 1NET ツールを使用する場合

2. 1NET ツールを使用する場合の概要

NET ツールを使用し IRS から PostgreSQL に対象データを移行する。

2. 2 操作内容

⑦ プロジェクトの作成：
プロジェクトを作成する

⑧ テーブルの準備：
データ移行元の IRS とデータ移行先の PostgreSQL に接続し、データベース、テーブル情報を取得する。

⑨ 抽出対象の設定：

全てのカラムを抽出対象に設定する。

⑩ カラム間の対応関係の設定：

データ移行元テーブルとデータ移行先テーブルのカラムの対応を設定する。

⑪ データ移行の開始：

5.1:HadoopAPI を用いた抽出の開始

5.2:HadoopAPI を用いた抽出の終了

5.3:変換・挿入の開始(HDFS からの読み込み、変換実行、データの挿入)

5.4:変換・挿入の終了

⑫ データ移行完了の確認:

データ移行が完了したことを確認する

2.3 結果

作業	操作完了時の 0.スタートからの経過時間(分:秒)
0.スタート	0
1.プロジェクトの作成	4 秒
2.テーブルの準備	55 秒
3.抽出対象の設定	1 分 12 秒
4. カラム間の対応関係の設定	1 分 30 秒
5. データ移行の開始	1 分 36 秒
5.1HadoopAPI を用いた抽出の開始	1 分 36 秒
5.2HadoopAPI を用いた抽出の終了	2 分 26 秒
5.3 変換・挿入の開始	2 分 26 秒
5.4 変換・挿入の終了	4 分 52 秒
合計	4 分 52 秒

(経過時間にはコマンドを打ち込む時間などを含む)

所要時間

作業	かかった時間
1～4 データ移行の準備	1 分 30 秒
5.1 HadoopAPI を用いた抽出	50 秒
5.2 変換挿入	2 分 26 秒

2.4 NET ツールを使用する場合まとめ

プログラミングや IRS と PostgreSQL のターミナル型フロントエンドを使用する必要がなく、NET ツールのみで ETL 作業を実行することができた。ストレージサーバの URL とトランザクションサーバの URL を入力するだけで、HadoopAPI を用いることができた。今回の実験は NET ツールの習熟者が行ったため、「1～4 データ移行の準備」は 1 分 30 秒で終了した。NET ツールに習熟していない人が行った場合でも、NET ツールは GUI を用いて操作することができるため、簡単にしかも短時間で ETL 作業が行えると考える。

3. 必要な知識一覧

・NET ツールなし：IRS の仕様, PostgreSQL の仕様, Java プログラミング, データベースに接続するための情報(データベースの URL, データベース名など)

・NET ツールあり：NET ツールの操作方法、データベースに接続するための情報(データベースの URL, データベース名など), HadoopAPI の設定情報(ストレージサーバの URL, トランザクションサーバの URL)

4. 比較

4.1ETL を行うまでの比較

NET ツール無しの場合はデータ抽出のプログラミングを行うために 20 分の時間がかかった。また、データベースの情報を入手する場合には IRS と PostgreSQL のターミナル型フロントエンドを使用する必要があり不便だった。

NET ツール有りの場合は、データ移行の準備としてテーブルの準備やカラム間の対応付けなどに 1 分 30 秒の時間がかかった。データベースやカラムの情報を見ながら GUI を用いて設定することができた。

4.2ETL 性能の比較

NET ツール無しの場合、のデータ抽出は、使用したプログラムの質によって所要時間が大きく変化することが考えられる。今回は IRS のマニュアルに記載されていたサンプルプログラムを参考にしたが、それでも 32 分の時間がかかった。データの挿入は PostgreSQL の COPY コマンドを使用した結果、高速にデータの挿入が行われた。

NET ツール有りの場合、データの抽出は、HadoopAPI を用いたため高速に行われた。データの挿入は NET ツールなしに比べて時間がかかった。

総評

第一イテレーションの際の実験よりもテストに使用したデータが 10 倍となり、HadoopAPI を用いた高速なデータの抽出による差が大きくなった。ユーザの技術的・知識的支援を行い、効率的な ETL 作業を行えるようにするという NET ツールの有効性が確認された。

第８章 納品フェーズ

本章では納品フェーズで作成した成果物について述べる。成果物は、Net ツール Eclipse プラグインファイル、ユーザズマニュアル、セットアップガイド、ツールを導入した Eclipse である。

第１節　Net ツールプラグイン

納品フォルダ内に作成した NET ツールの Eclipse プラグインファイル(jp.ac.tsukuba.cs.itsoft.team_net.irs2db.jar)を配置した。

第２節　ユーザズマニュアル

本節では、本ツールで行える操作を示したユーザズマニュアルの内容を記述する。納品フォルダ内のマニュアルフォルダにユーザズマニュアルの内容を示したファイルを配置した。

第９章　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。

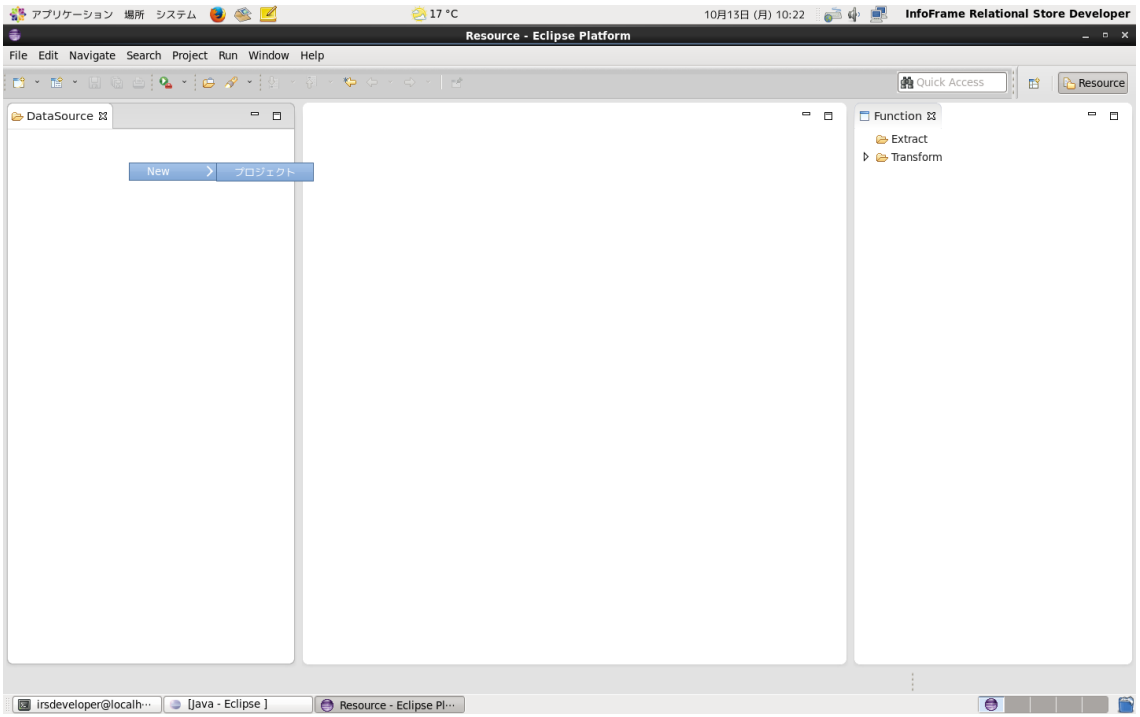


図 34　データソースウィンドウ内で右クリックする

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。

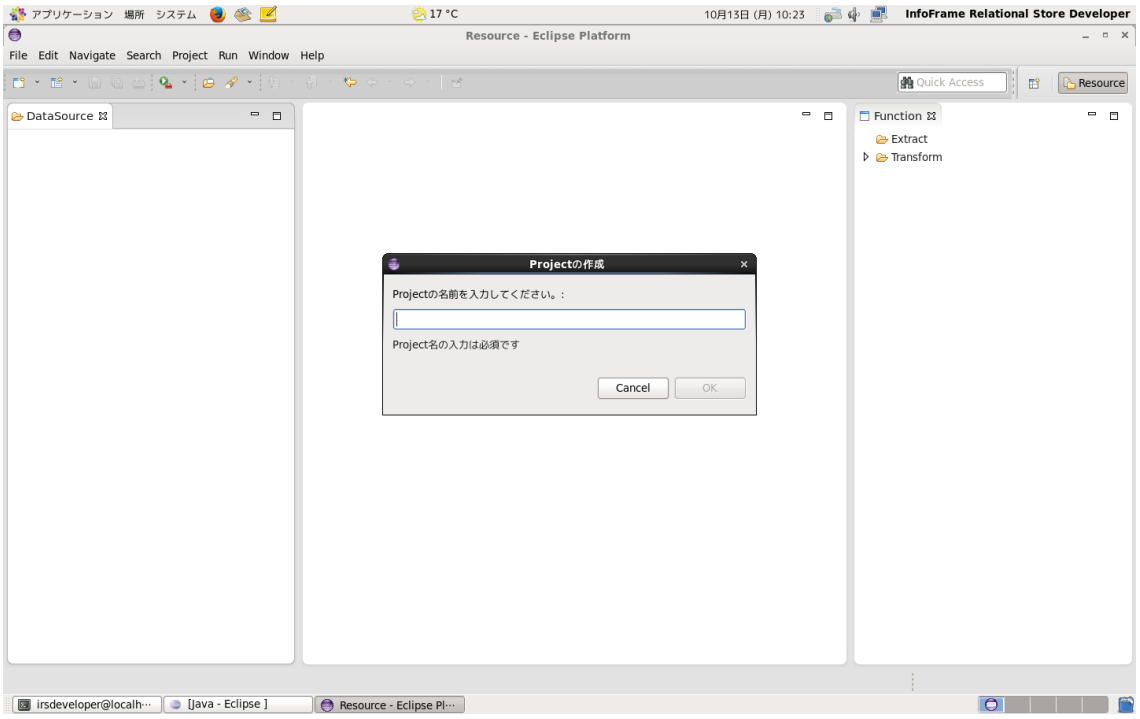


図 35　プロジェクト作成ダイアログ

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。

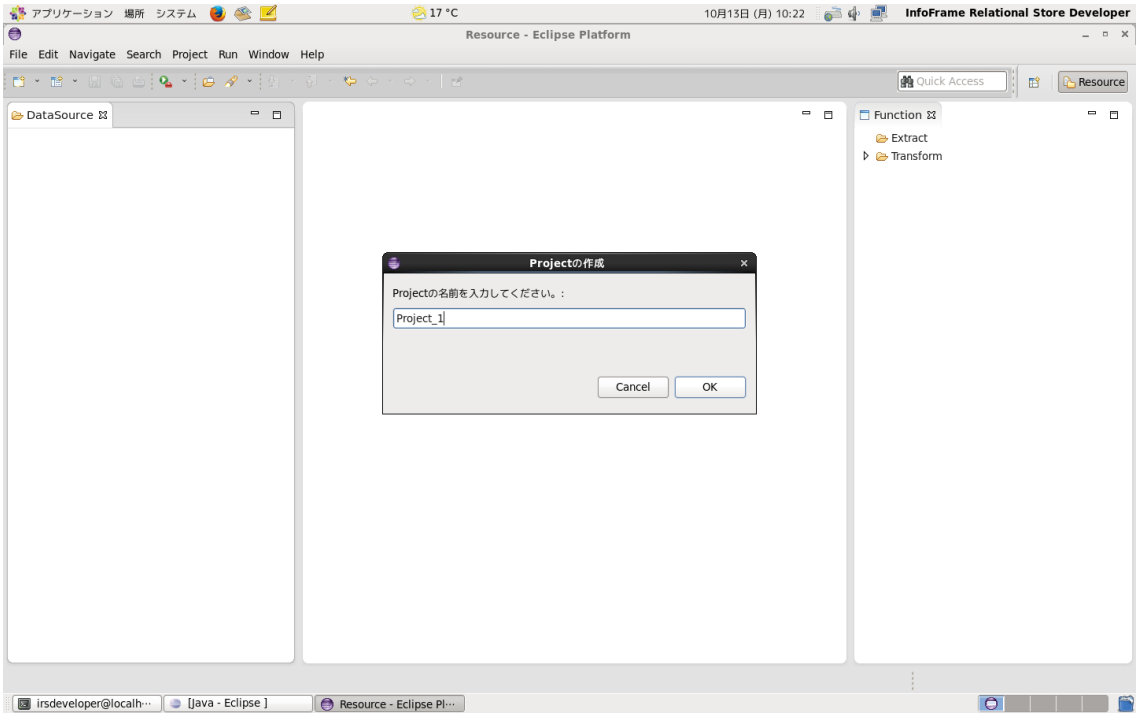


図 36　プロジェクト名の入力

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。

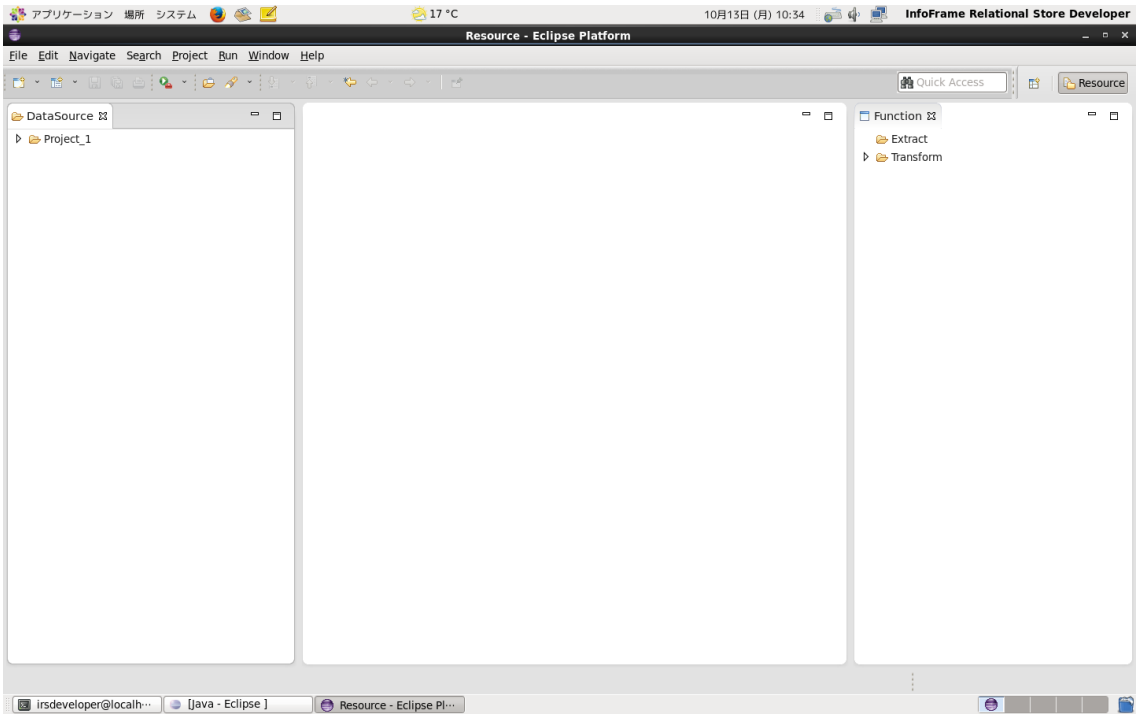


図 37 プロジェクト作成後

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。

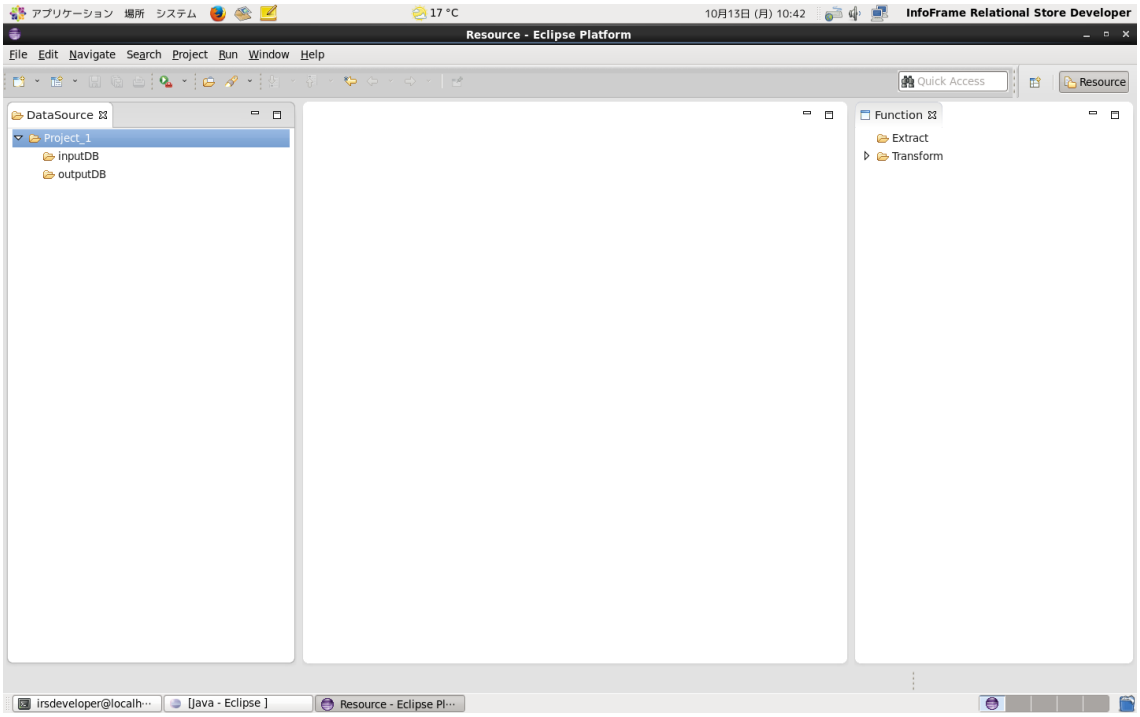


図 38　intputDB と outputDB の確認

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。

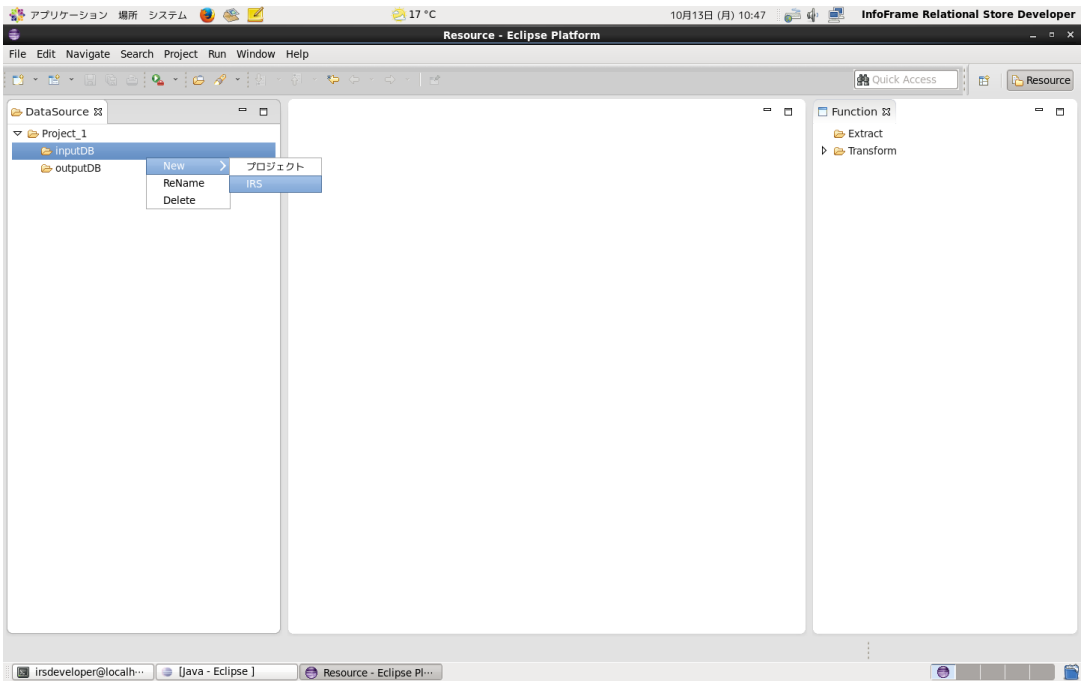


図 39　IRS を選択

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。

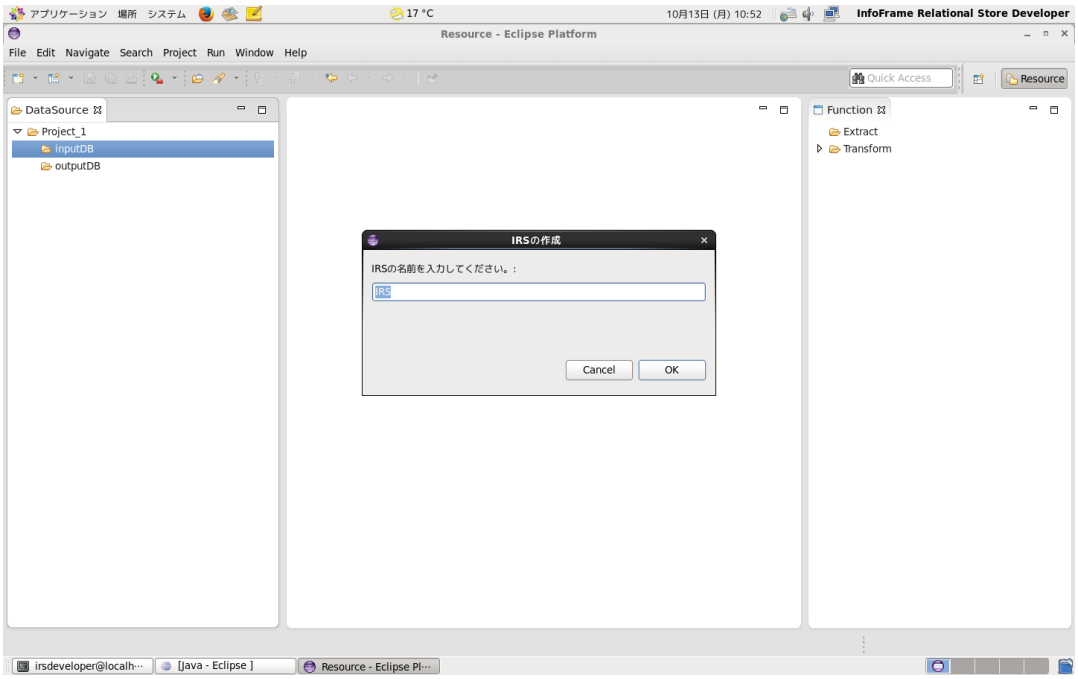


図 40 IRS の作成ダイアログ

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。

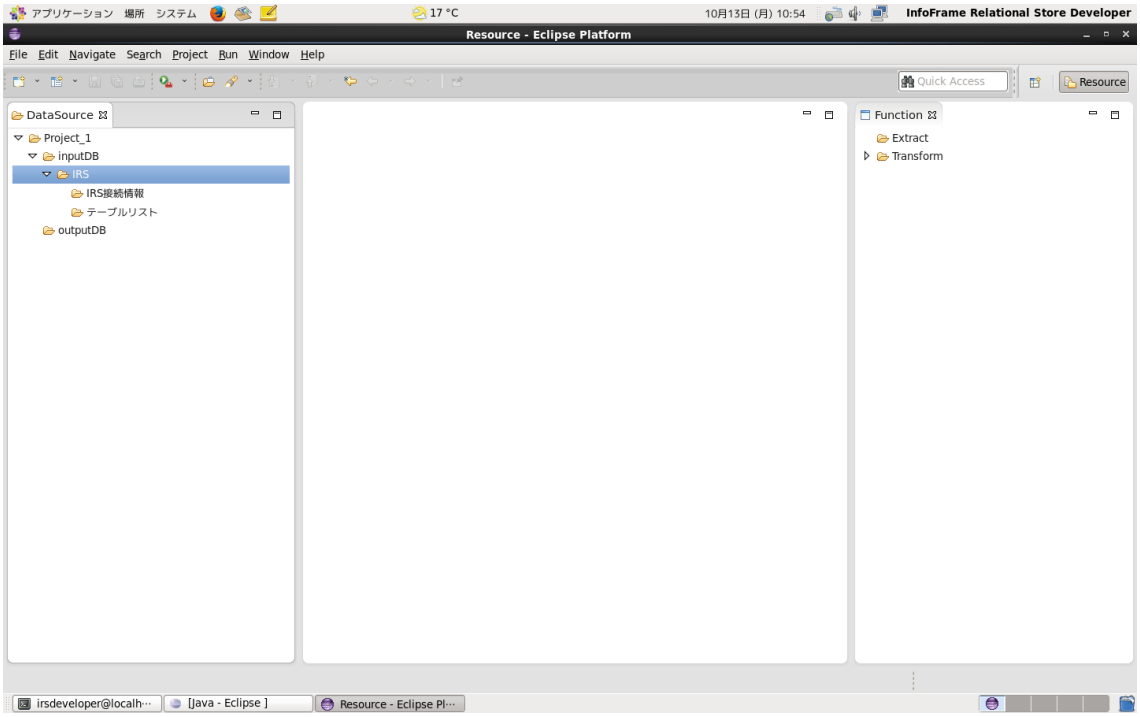


図 41 IRS 作成後の画面例

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。

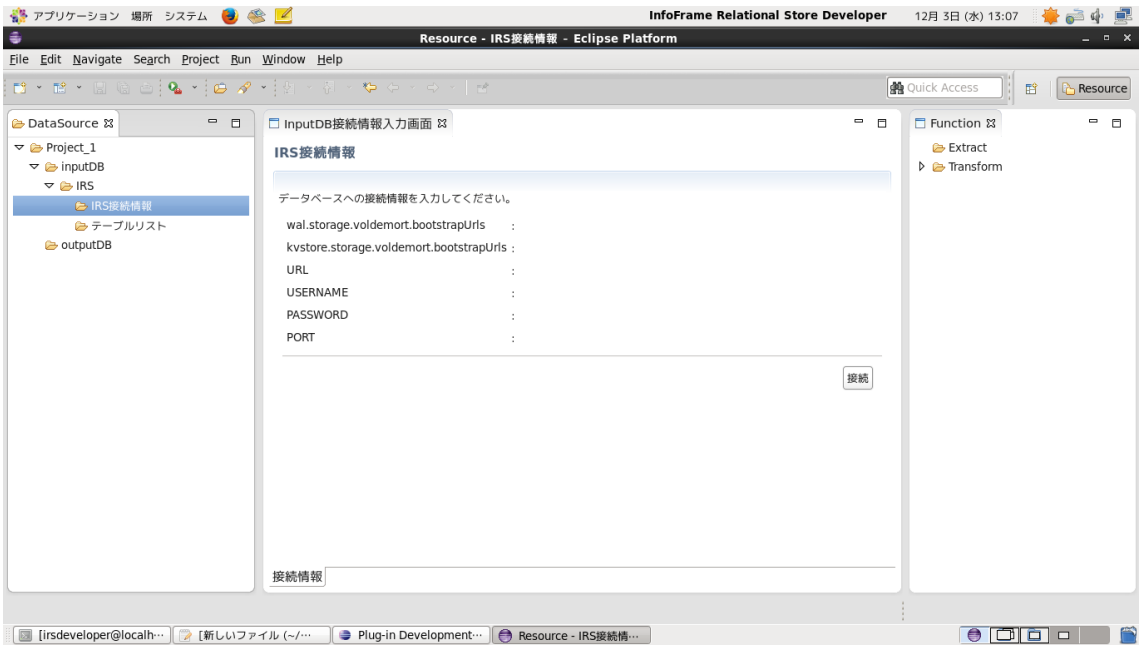


図 42　InputDB 接続情報入力画面

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

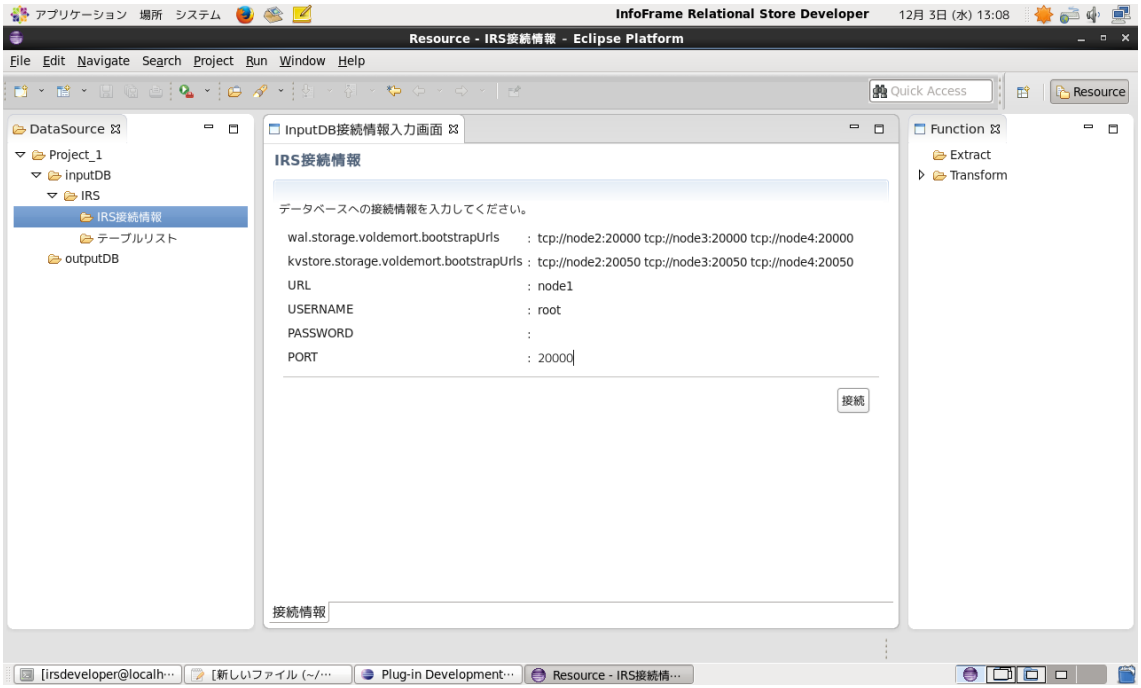


図 43 入力例

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。

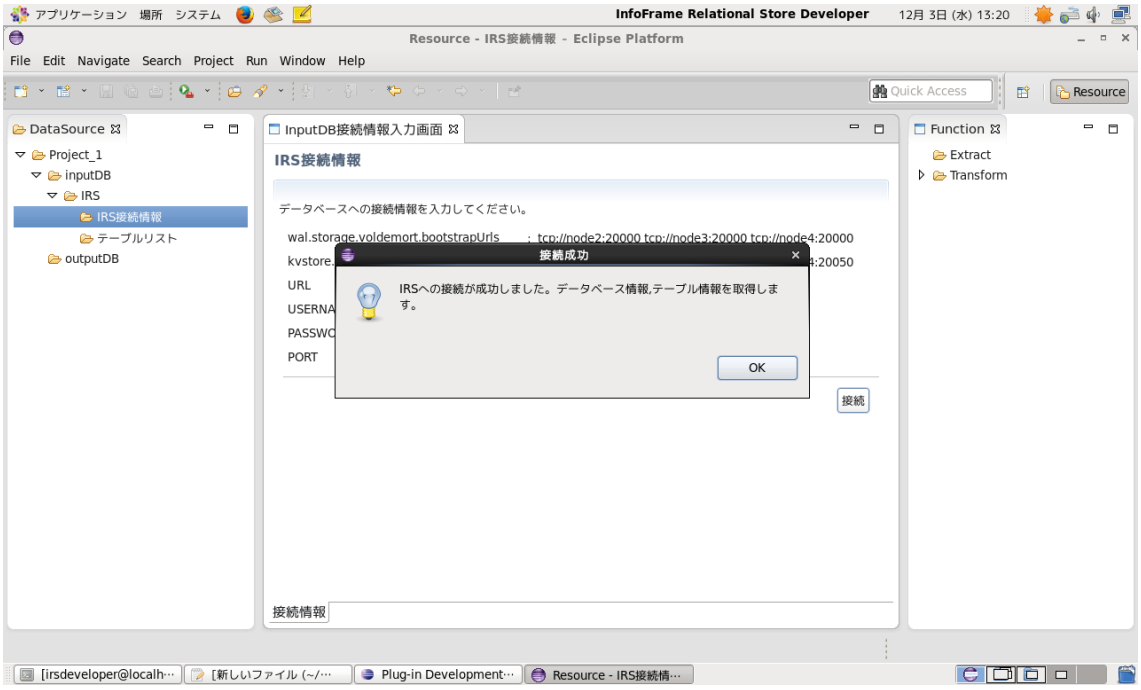


図 44 接続の成功

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。

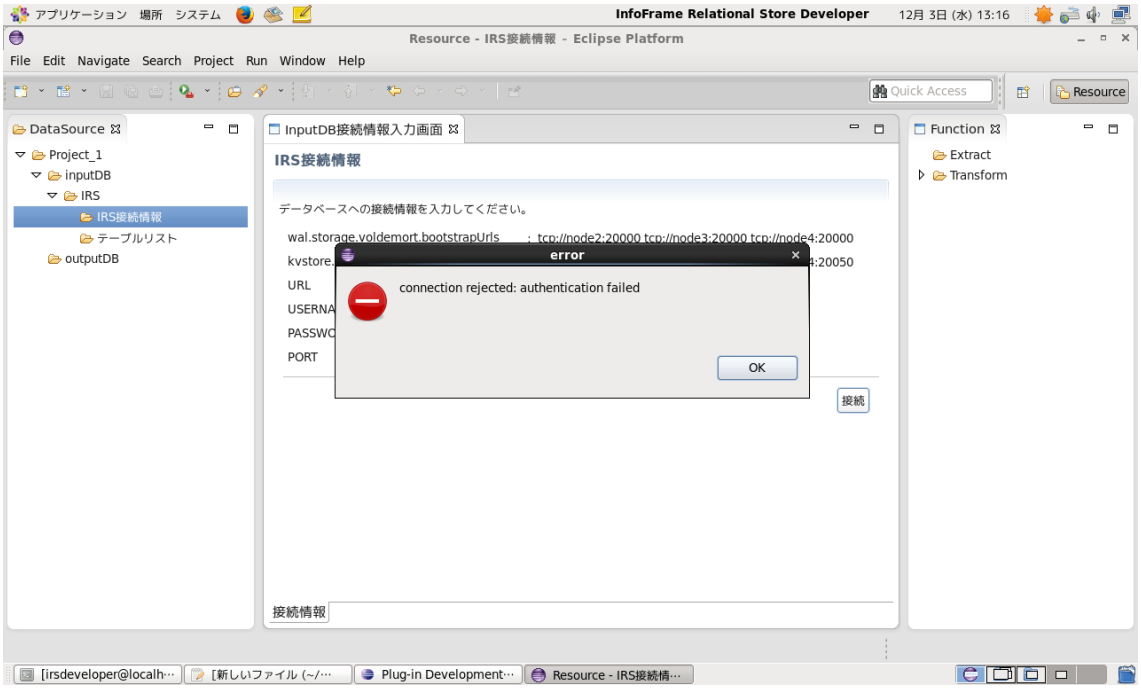
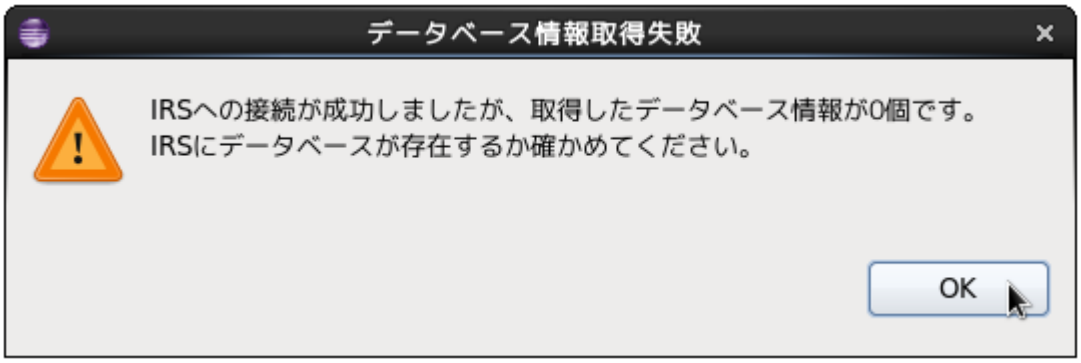


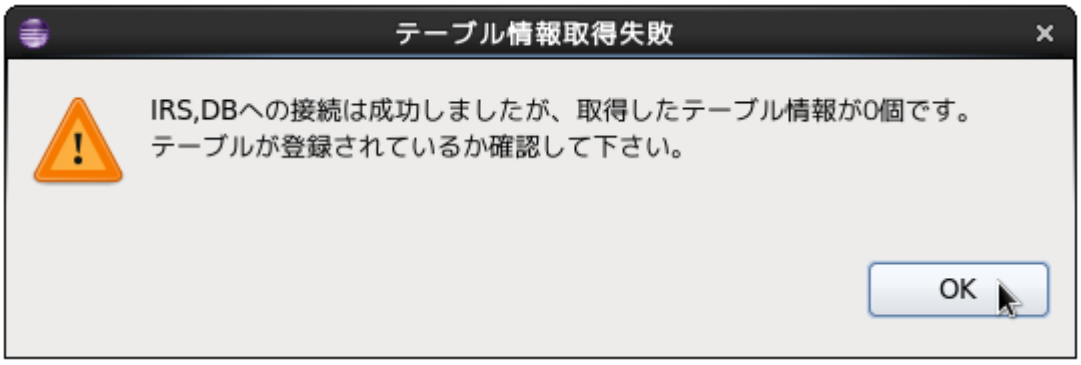
図 45 接続の失敗

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。

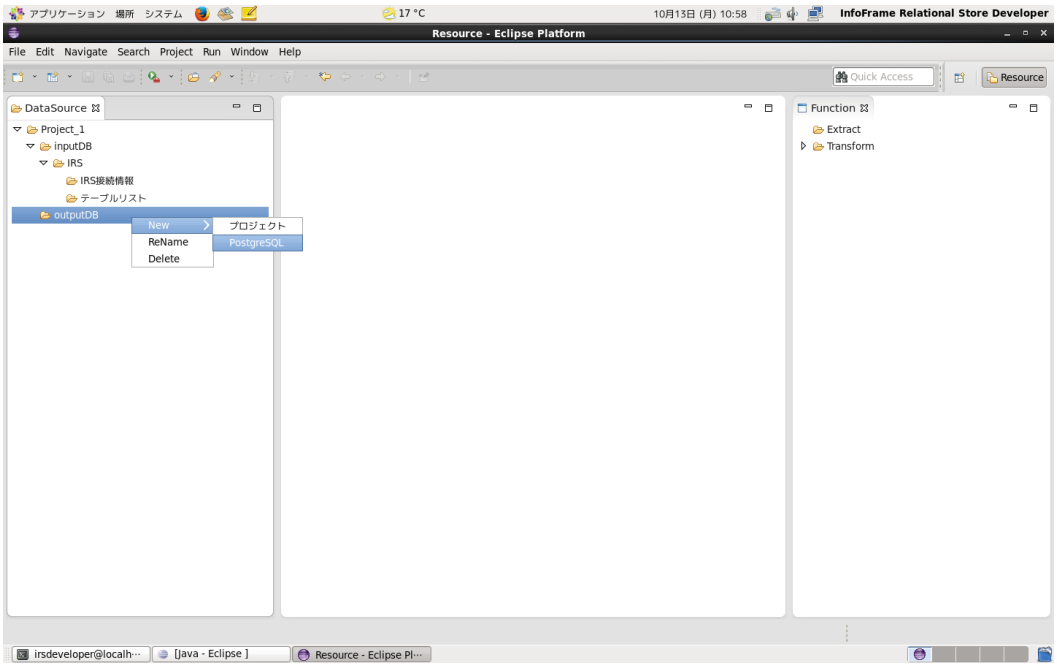


図 46 PostgreSQL を選択

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。

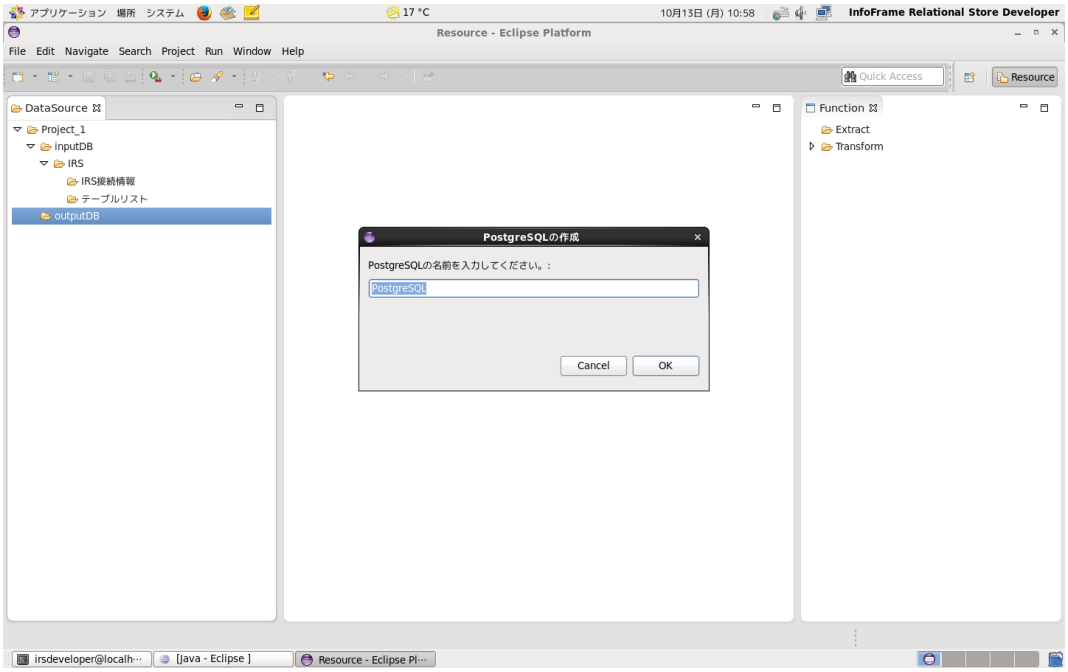


図 47 PostgreSQL 作成ダイアログ

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。

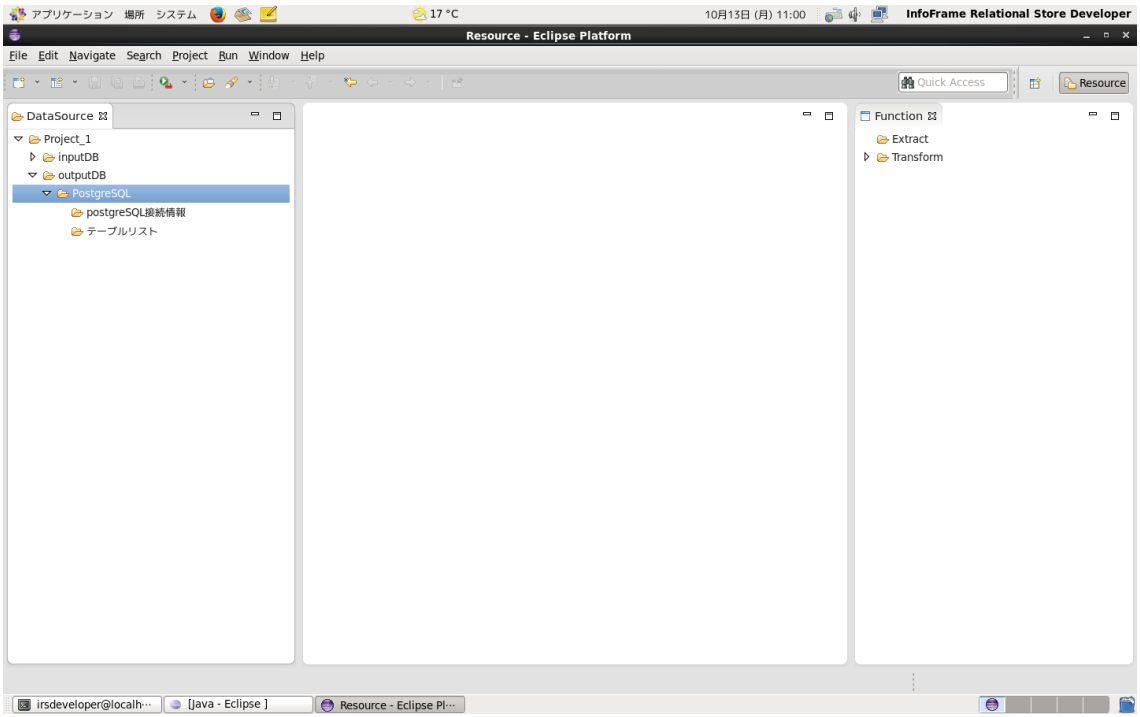


図 48 PostgreSQL 接続情報、テーブルリストの確認

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。

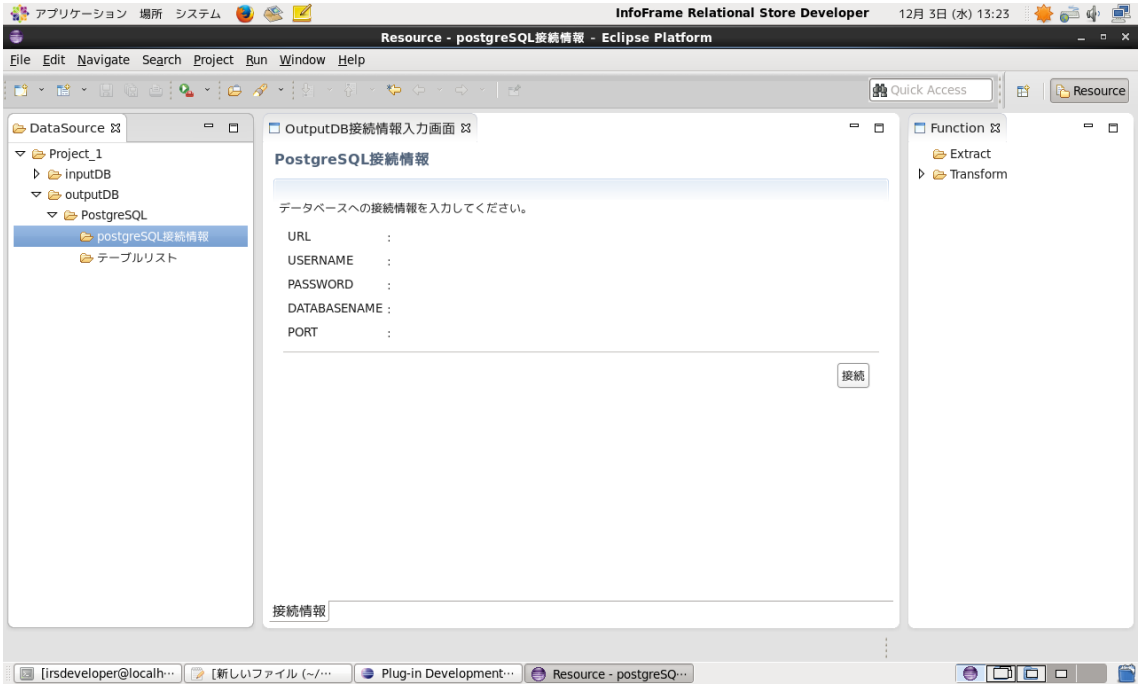


図 49　OutputDB 接続情報入力画面

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	

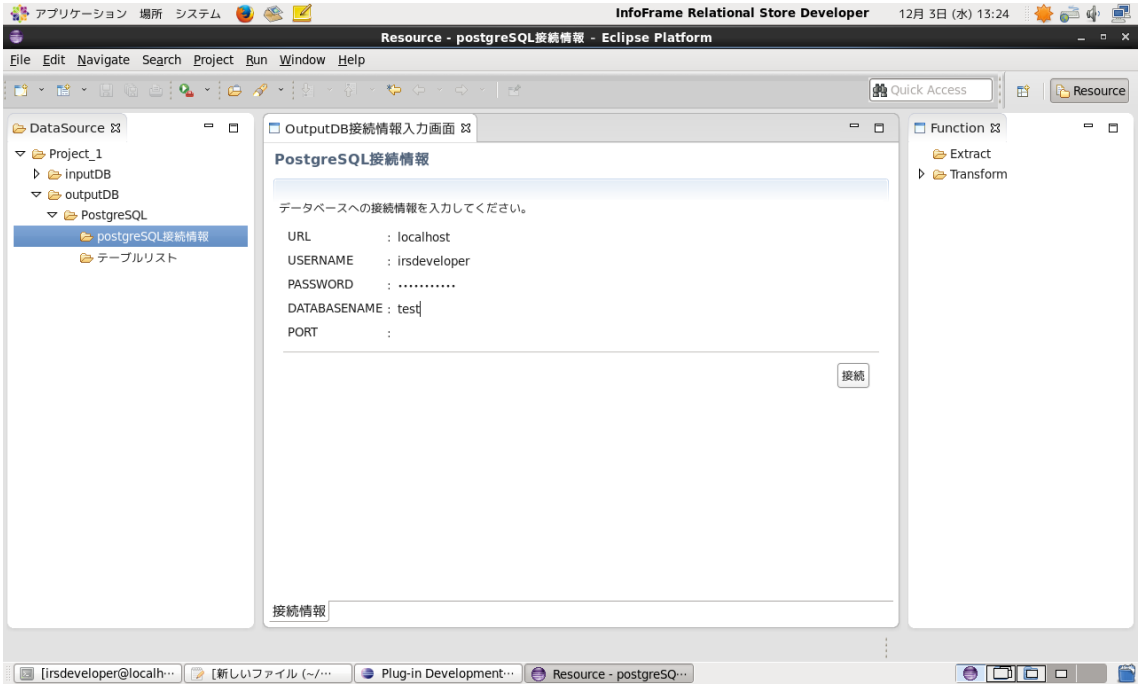


図 50　入力例

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。

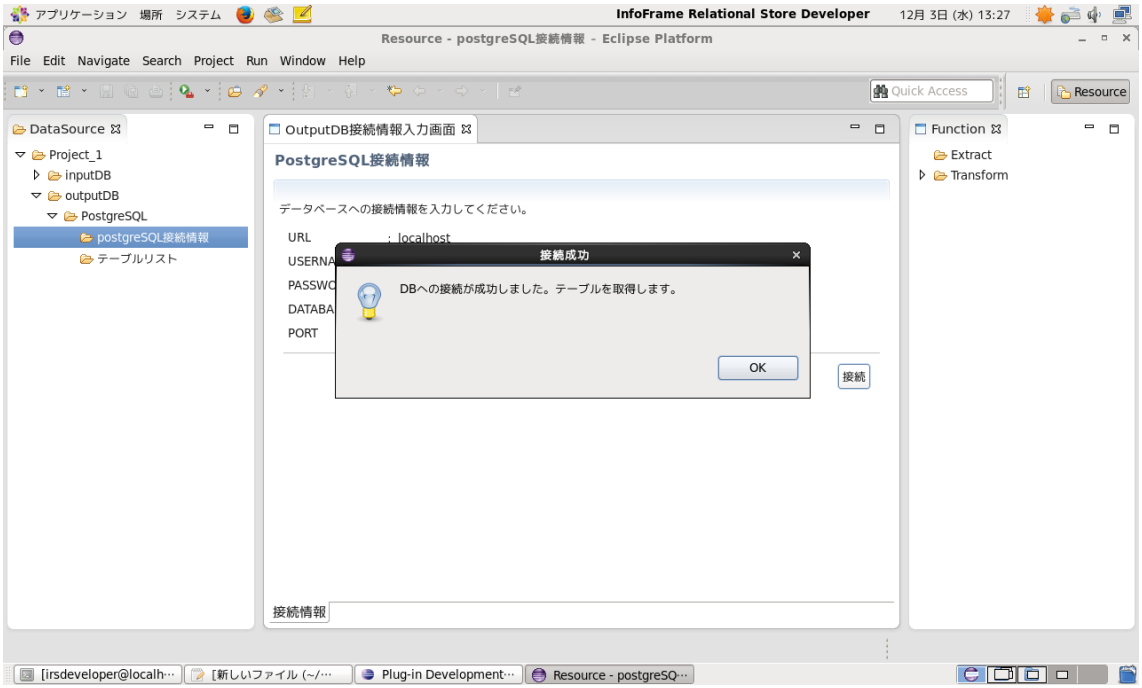


図 51 接続の成功

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。

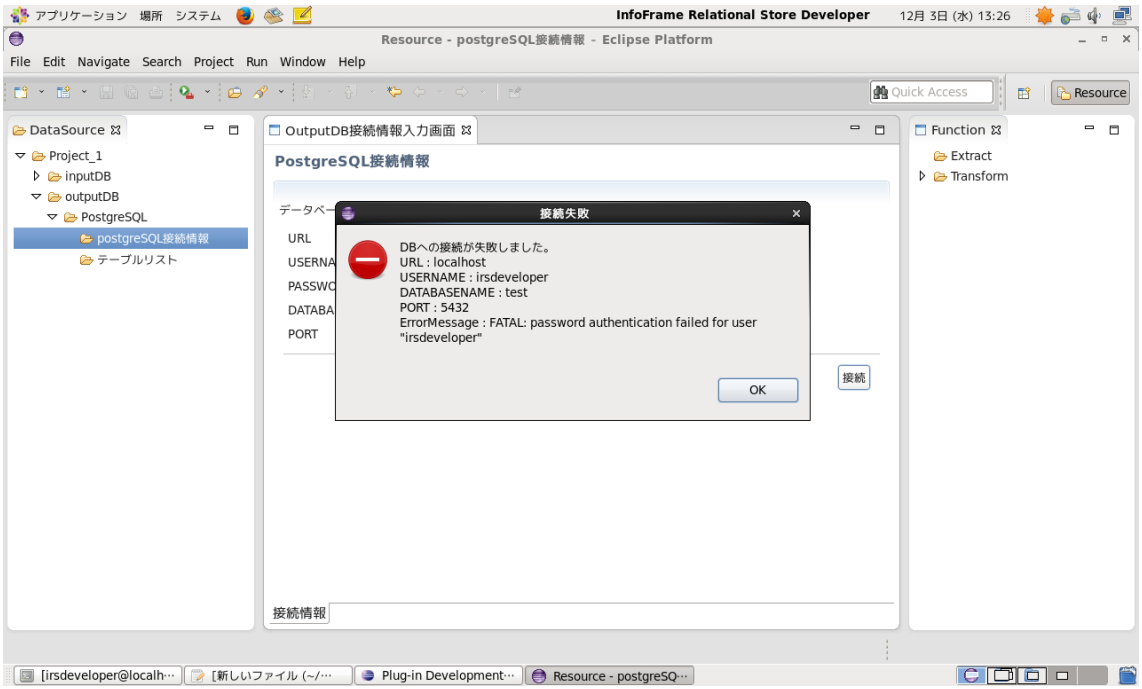


図 52 接続の失敗

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。

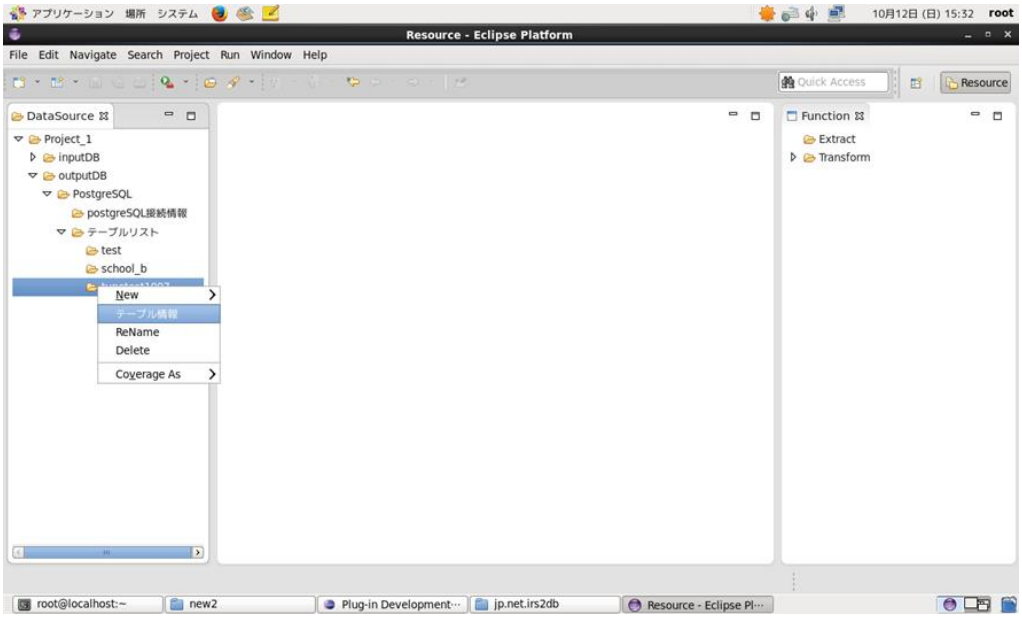


図 53　テーブル情報の選択

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。

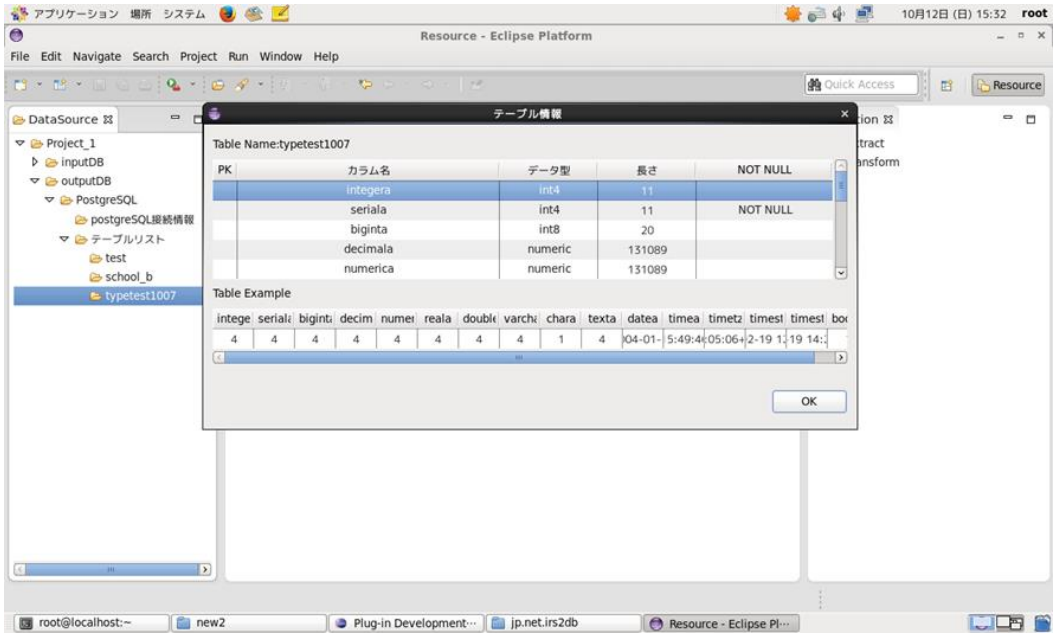


図 54　テーブル情報の確認

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。

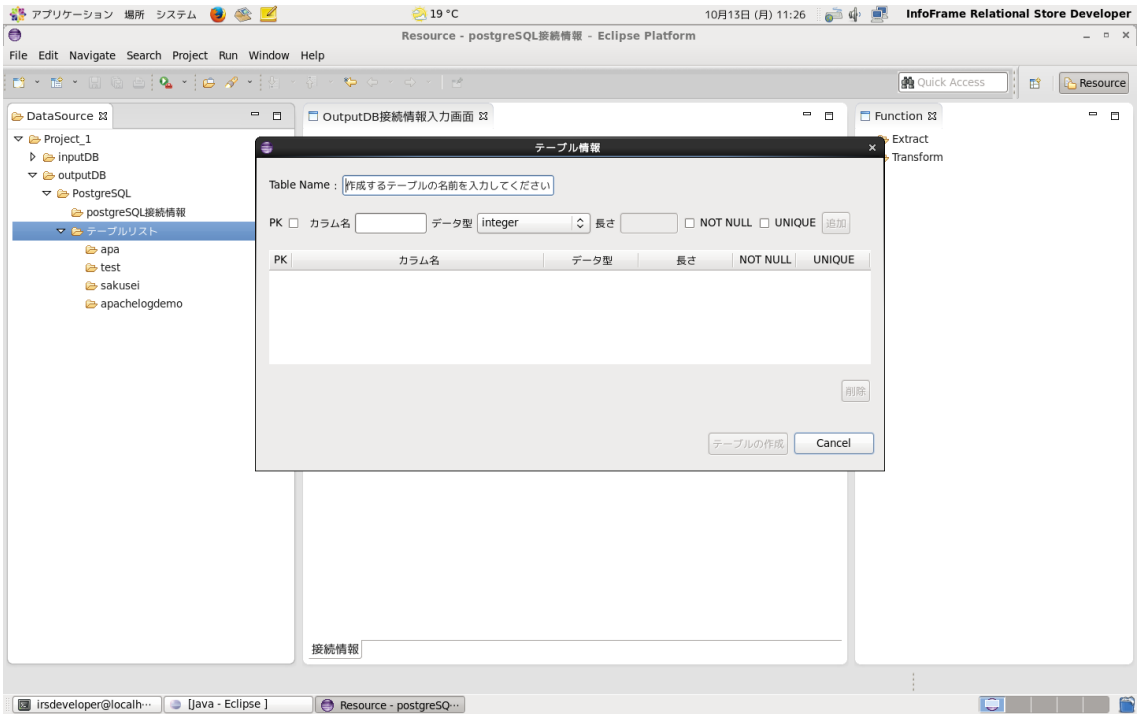


図 55　テーブル作成ダイアログ

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

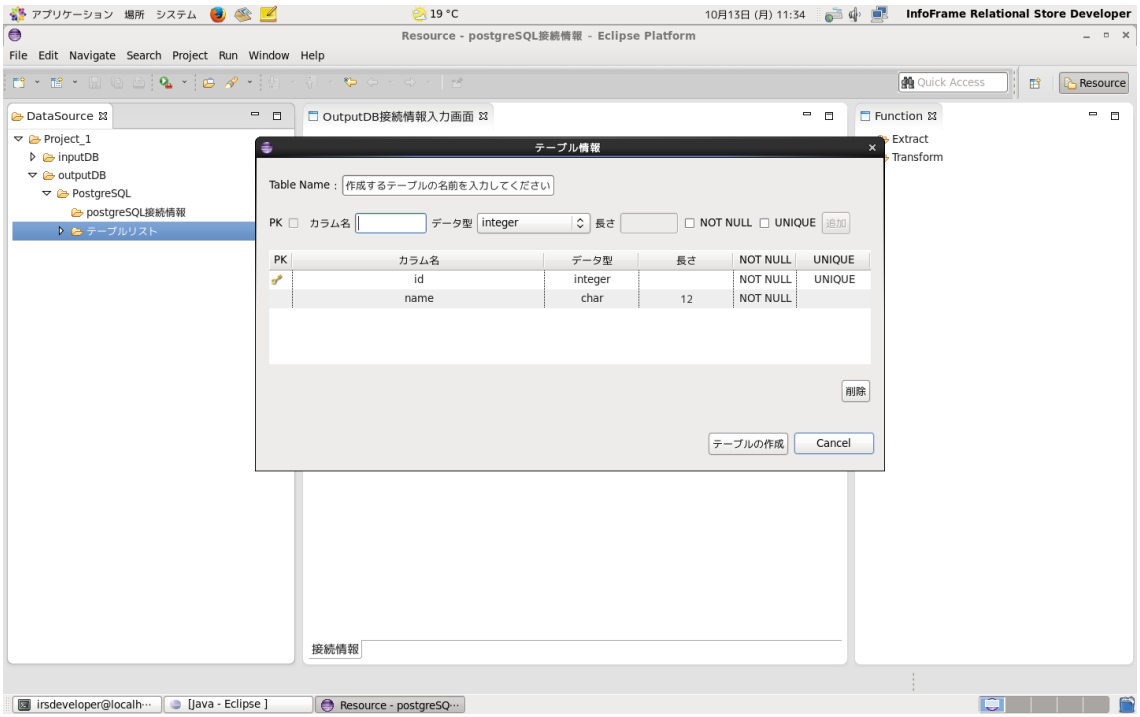


図 56 カラムの追加

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。

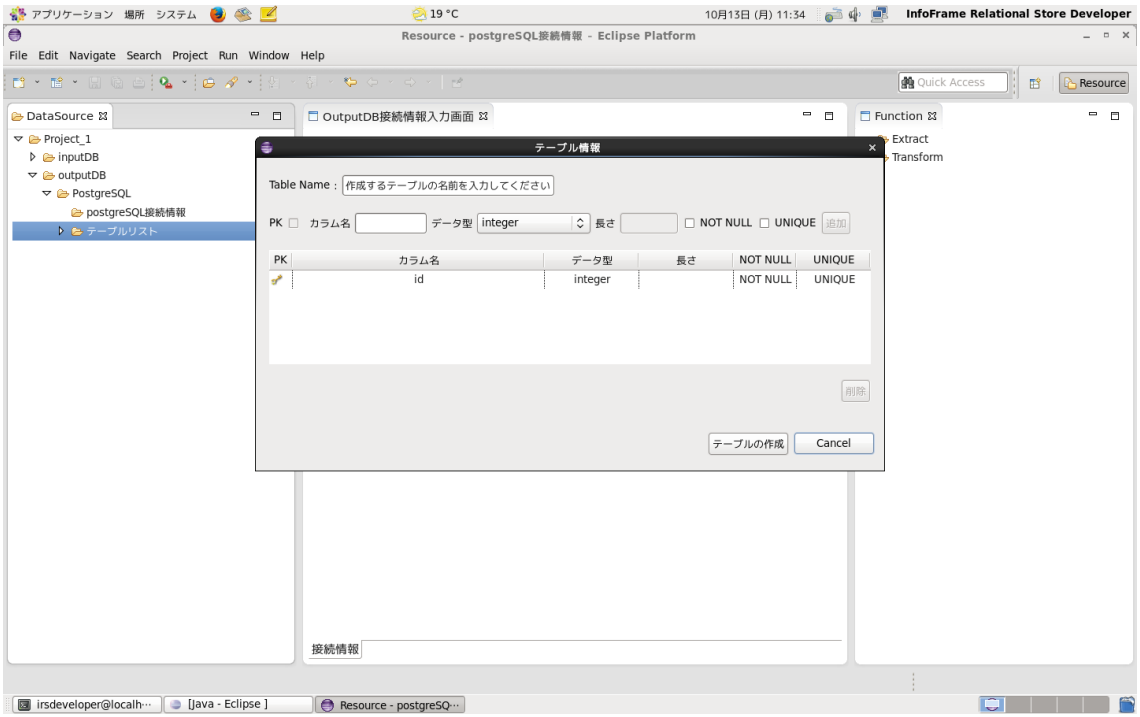


図 57 カラムの削除

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。

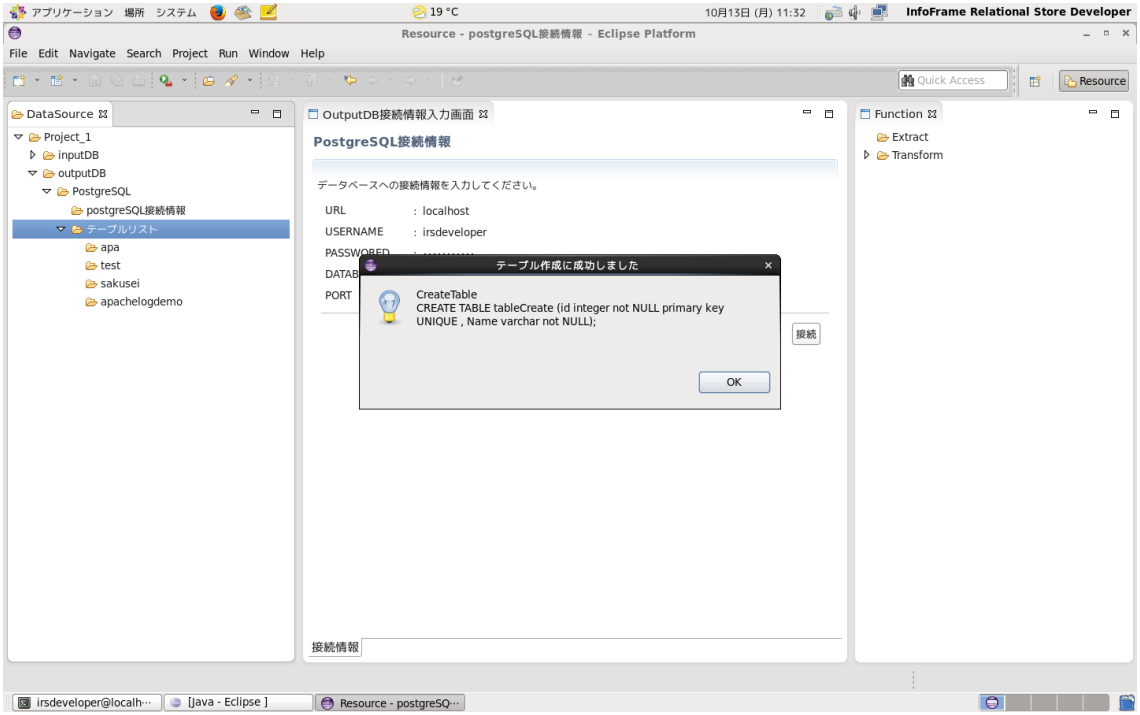


図 58 テーブル作成成功

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。

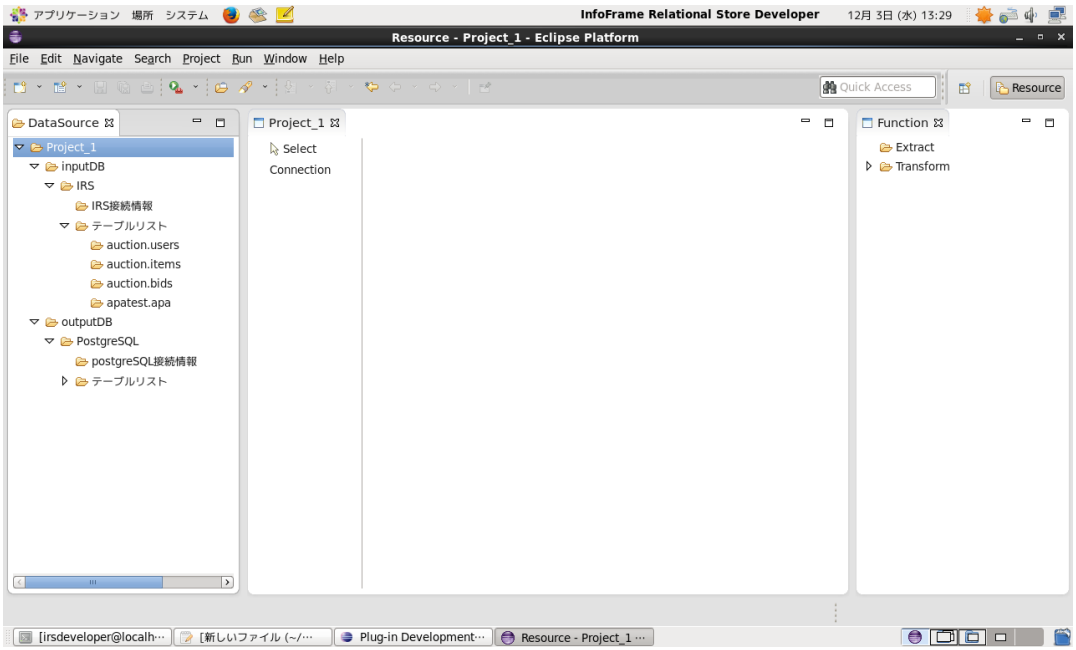


図 59 作業ウィンドウを開く

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。

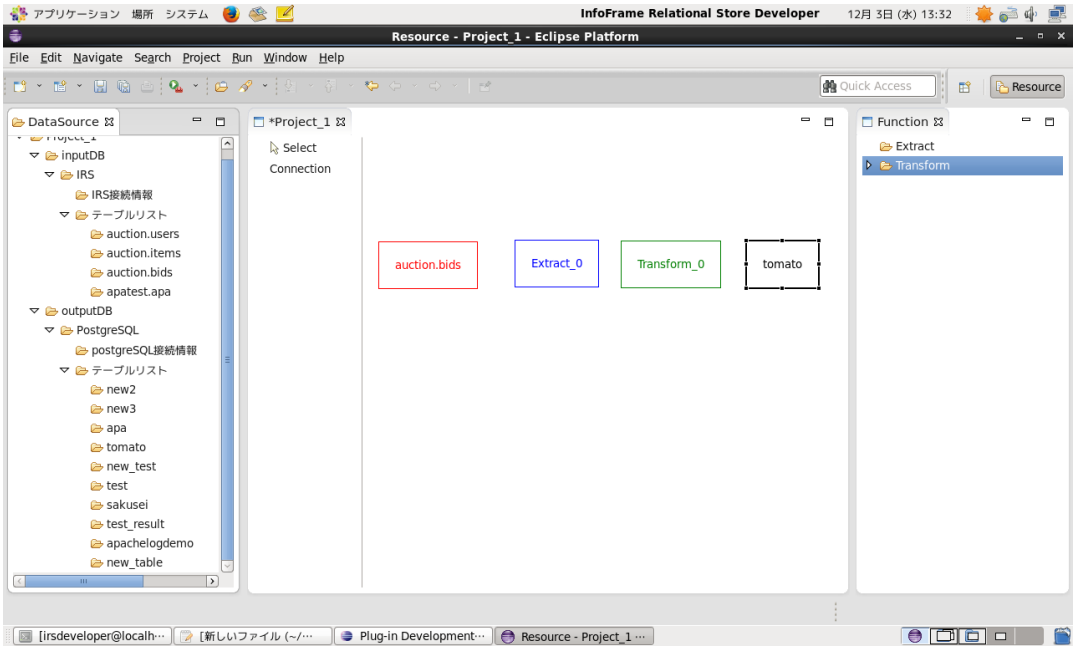


図 60　作業ウィンドウにドラッグアンドドロップする

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。

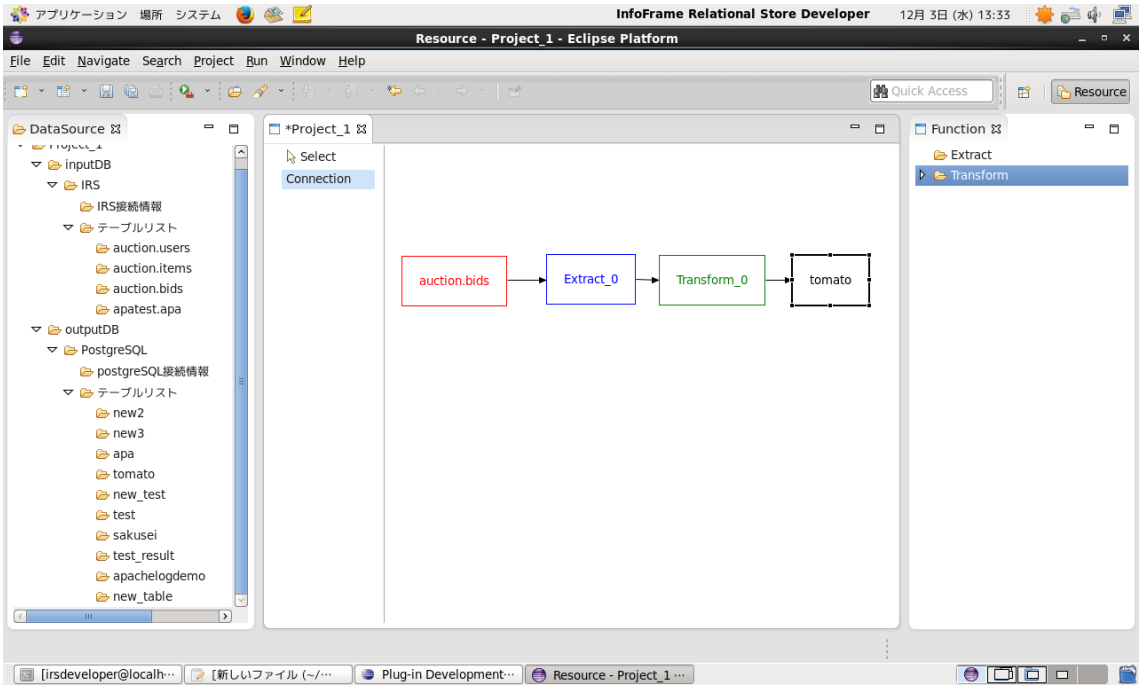


図 61 アイコンの紐付け

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。

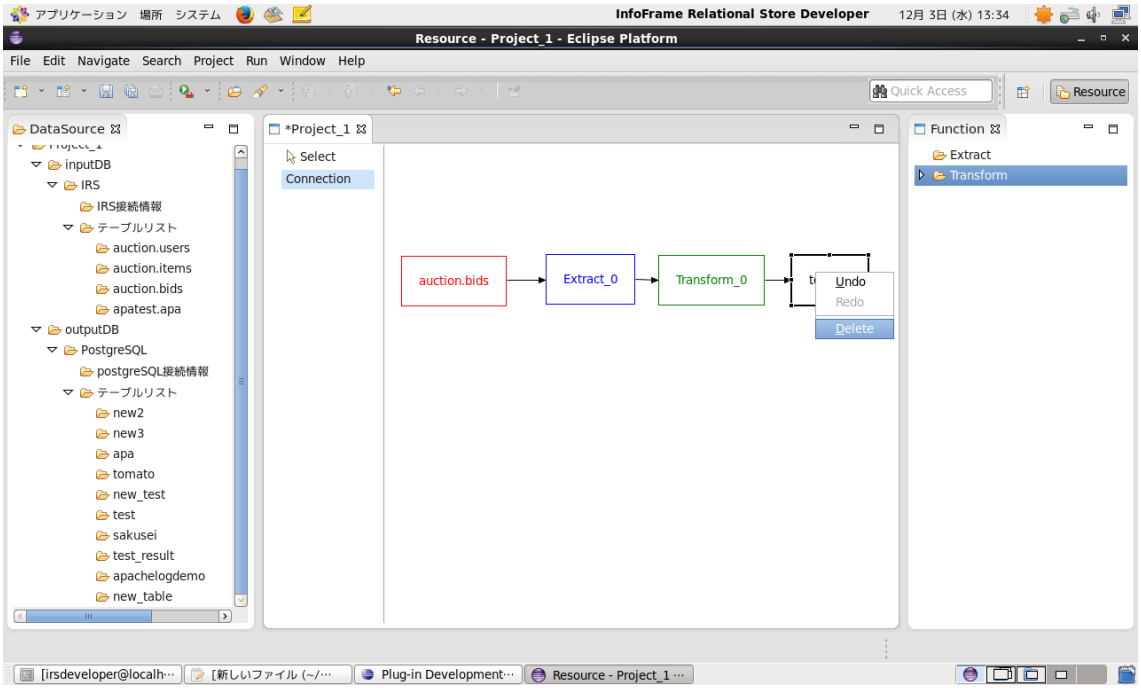


図 62 アイコンの削除（1）

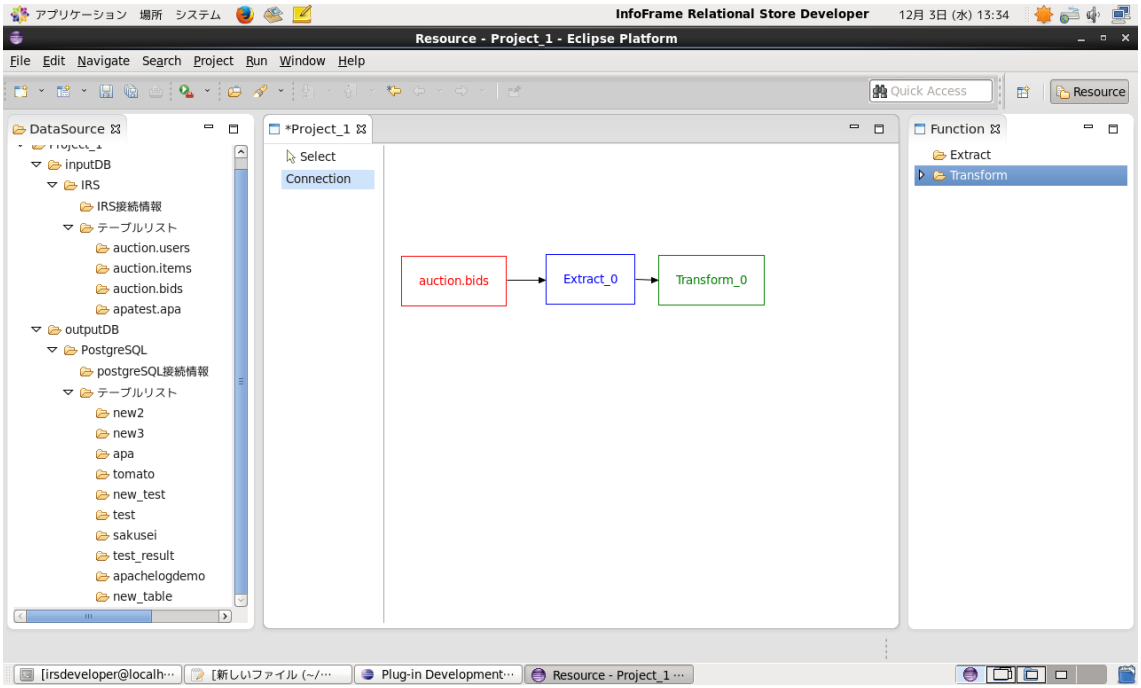


図 63 アイコンの削除（2）

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。

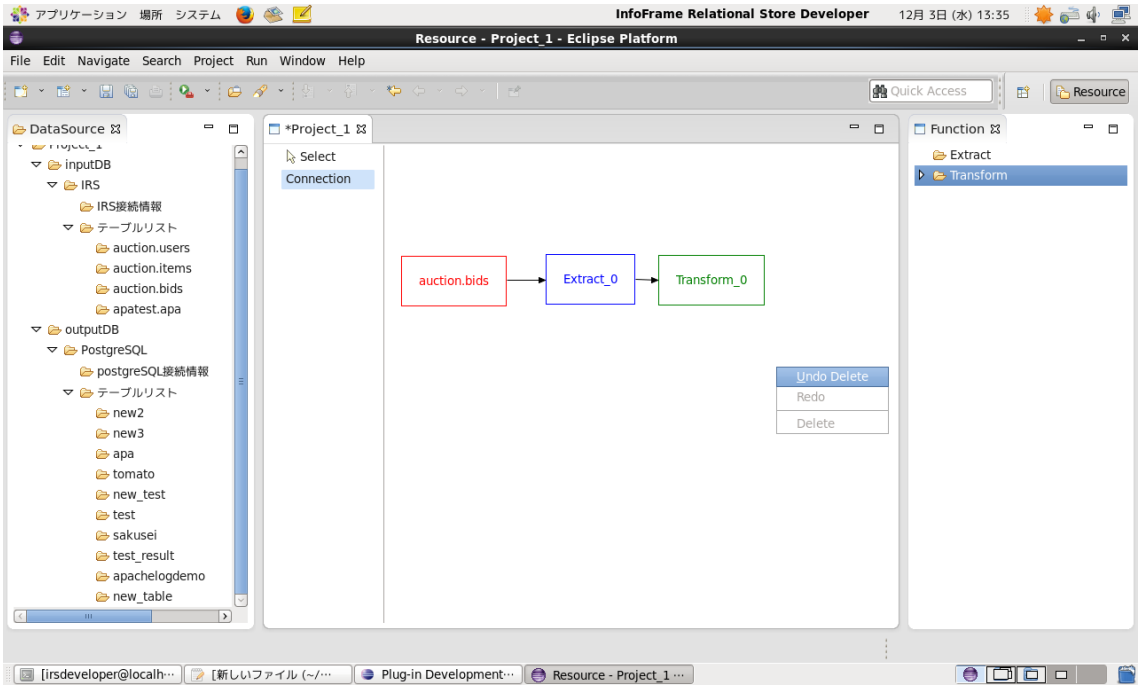


図 64 Undo 操作

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。

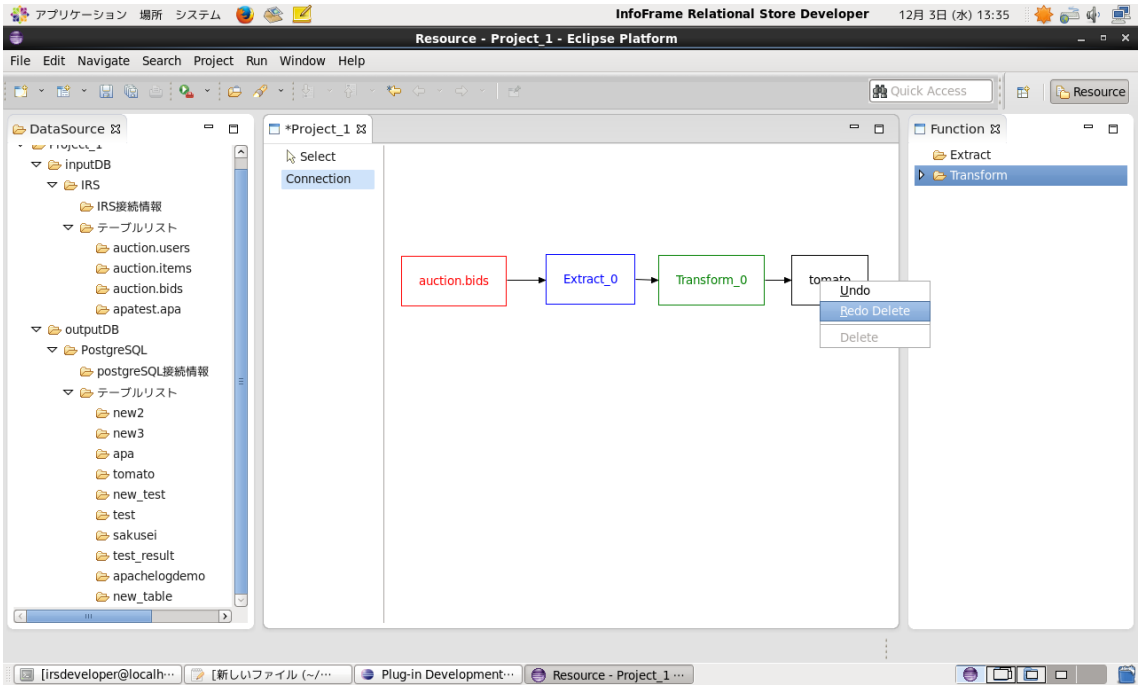


図 65 Redo 操作

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。

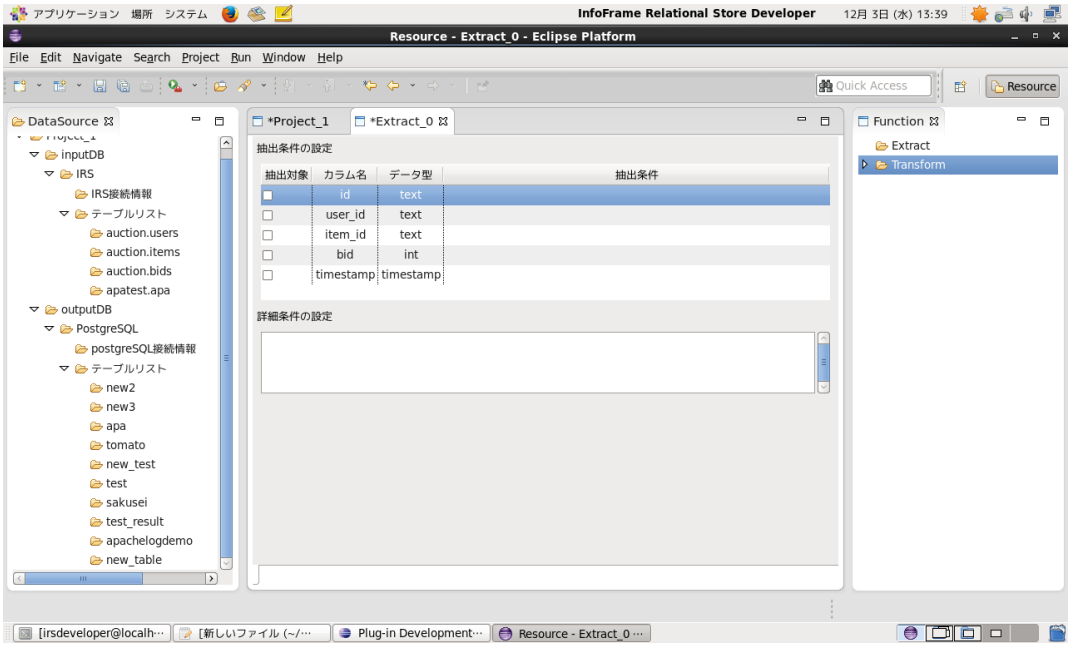


図 66 Extract ウィンドウ

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。

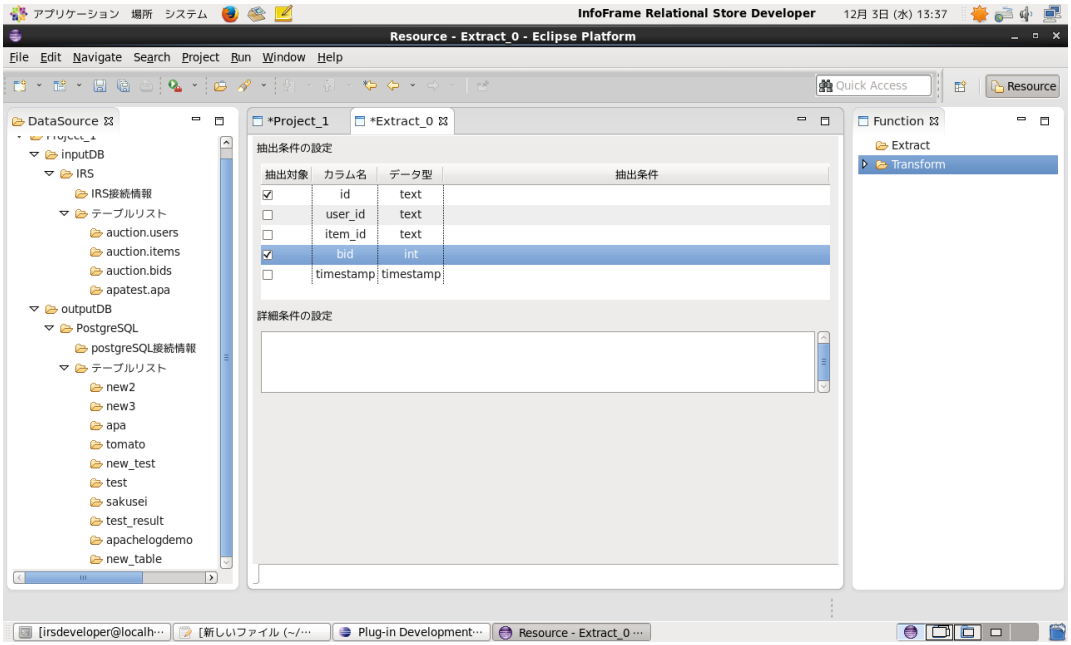


図 67 抽出対象とする

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。

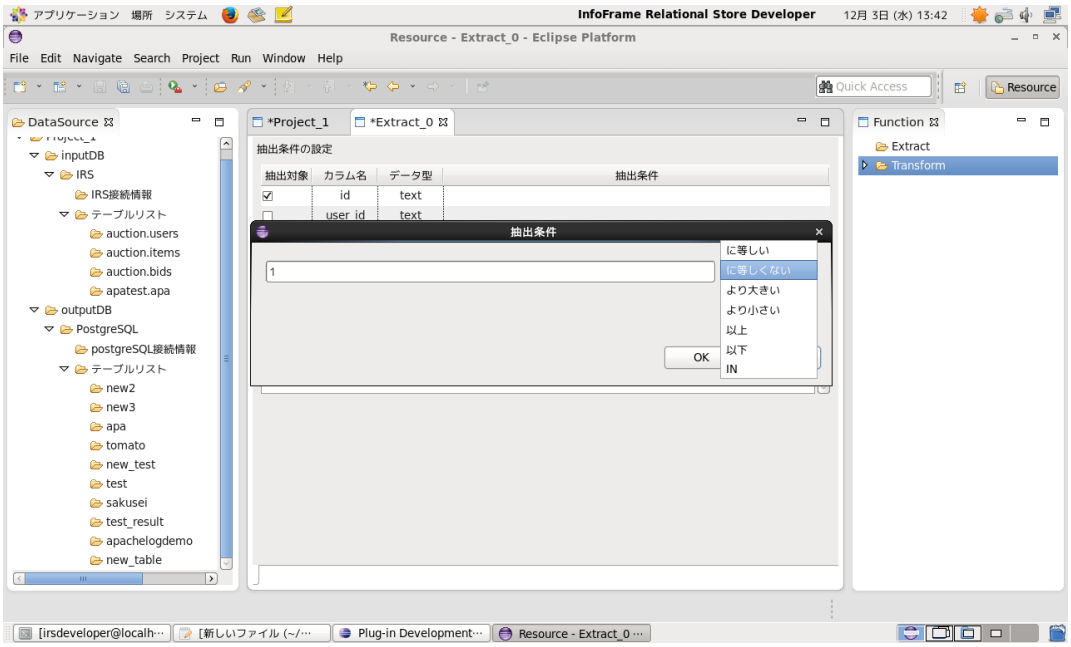


図 68 条件抽出設定画面

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。

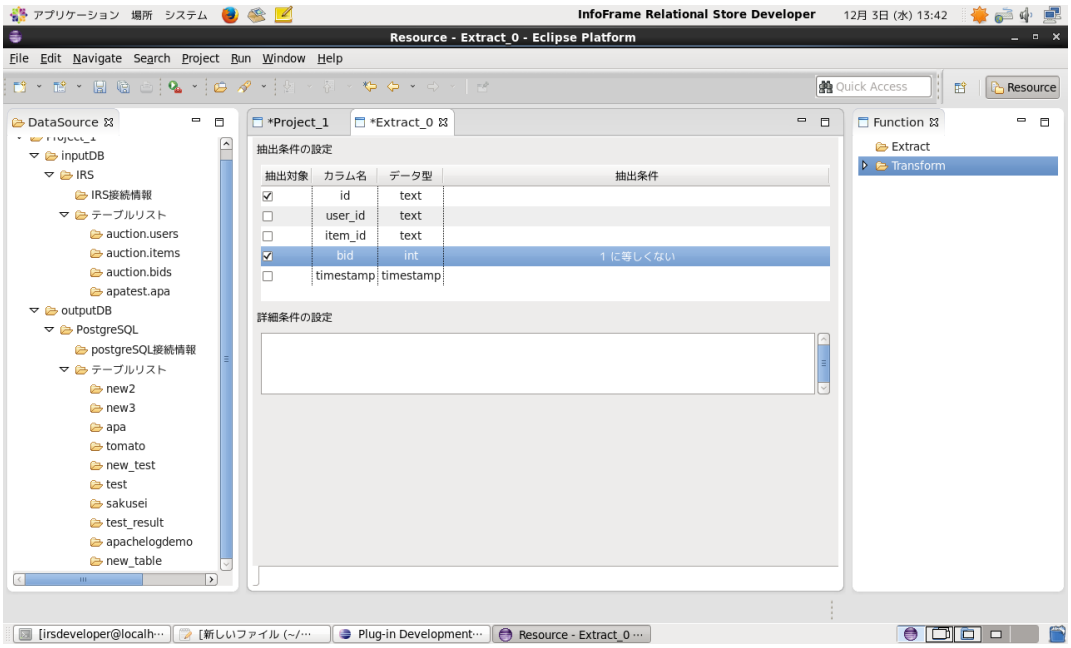


図 69 条件設定後

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。

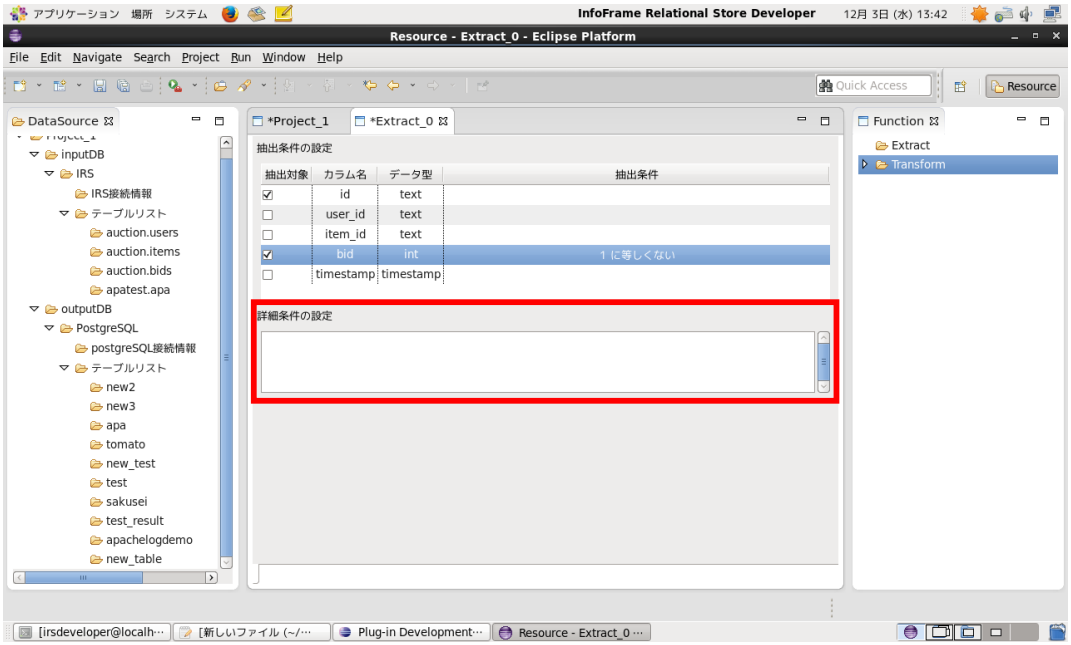


図 70 詳細条件の設定画面

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。

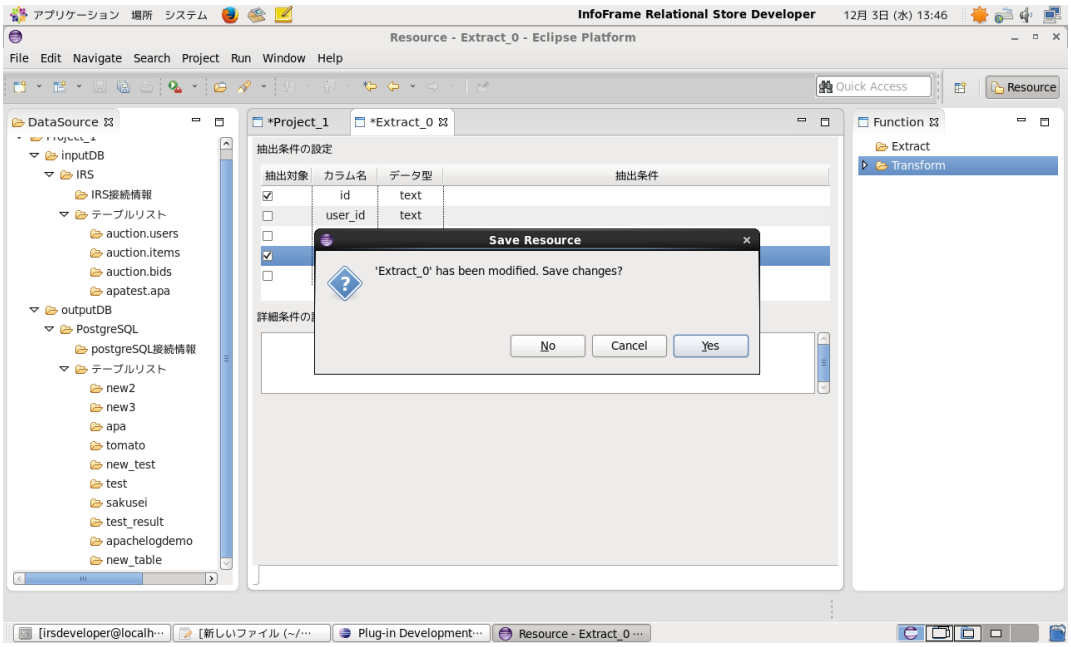
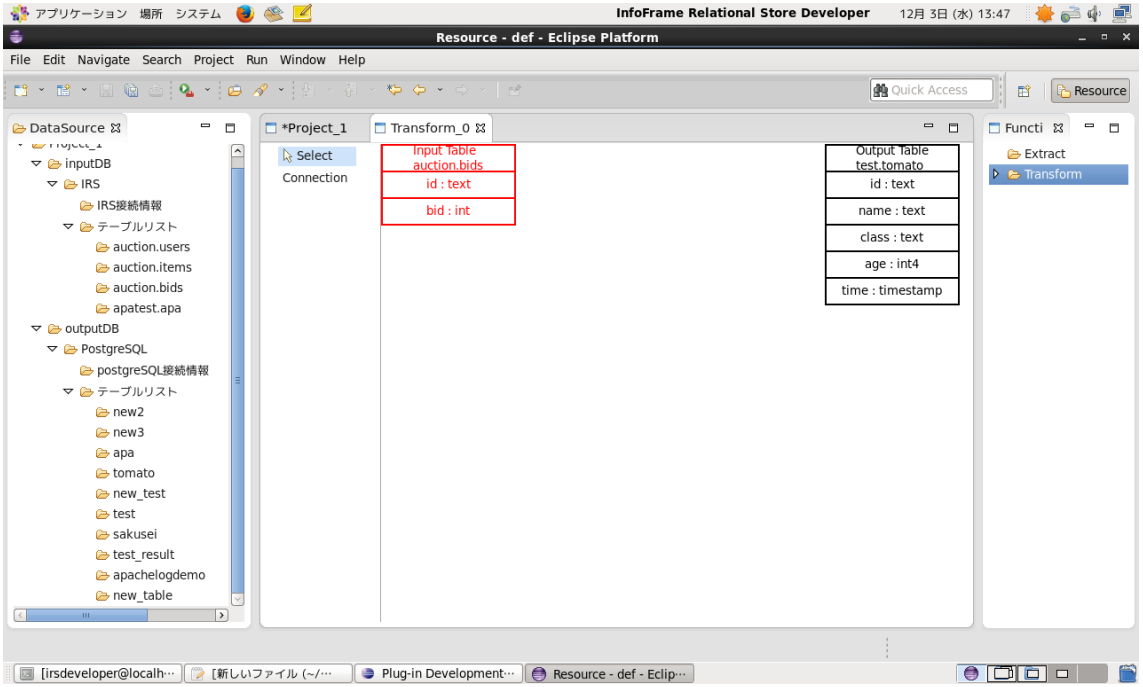


図 71 セーブ確認画面

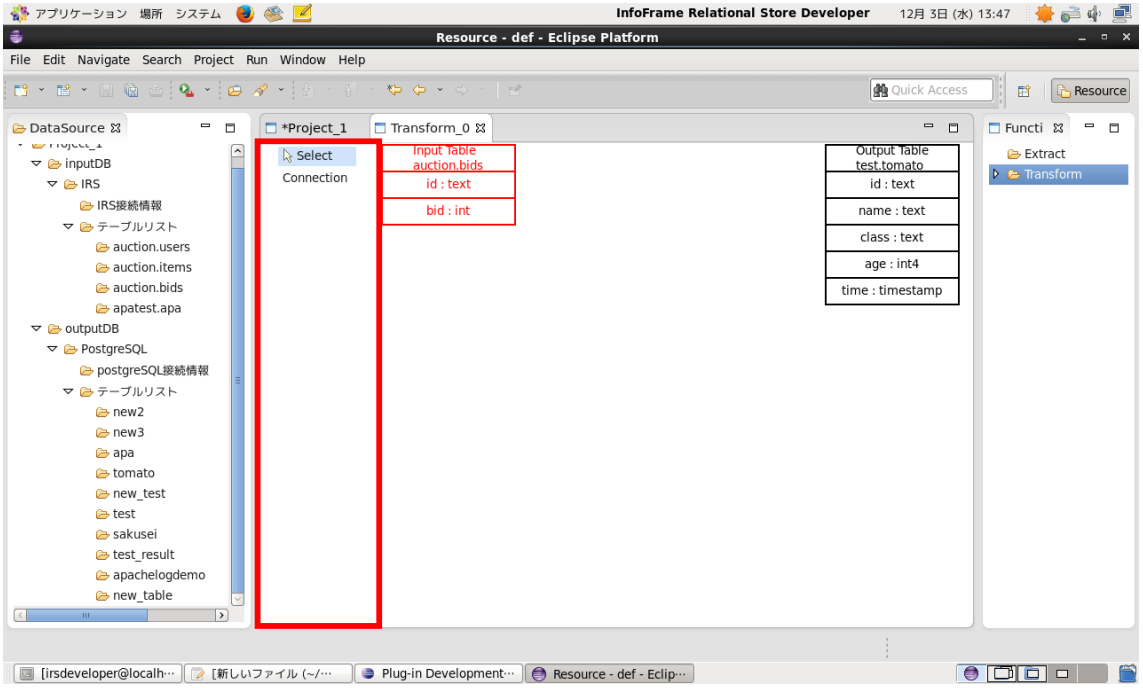
10. Transform 条件を設定する

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。



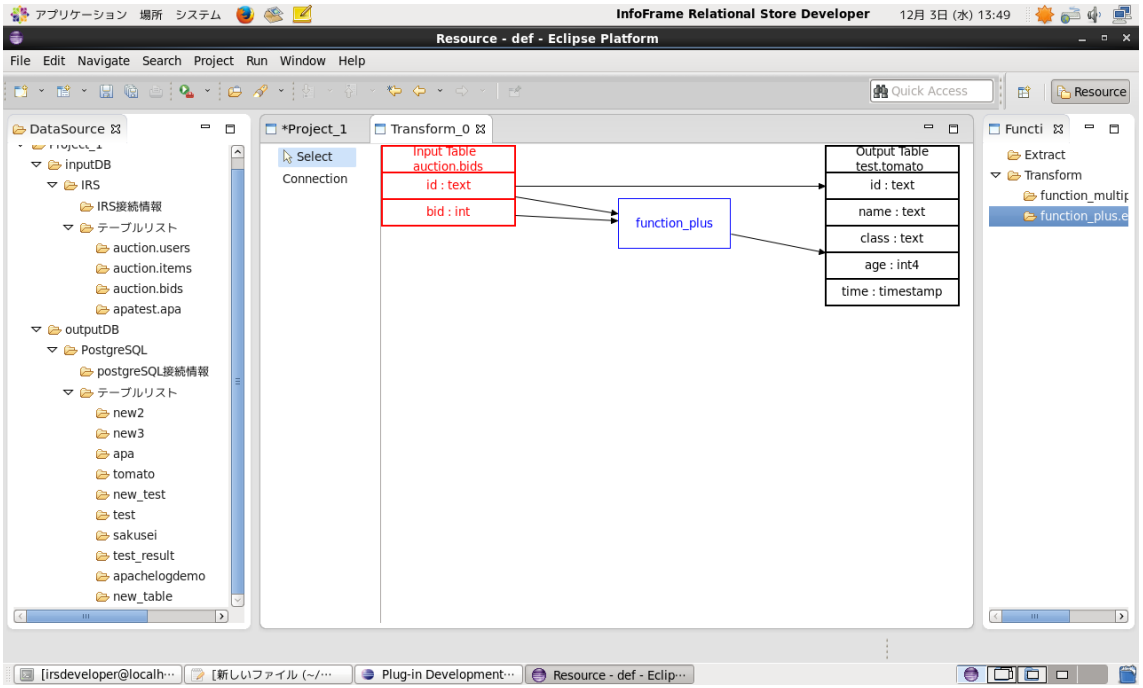
2. 作業ウィンドウの左側に「Select」と「Connection」というツールがあります。

「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。

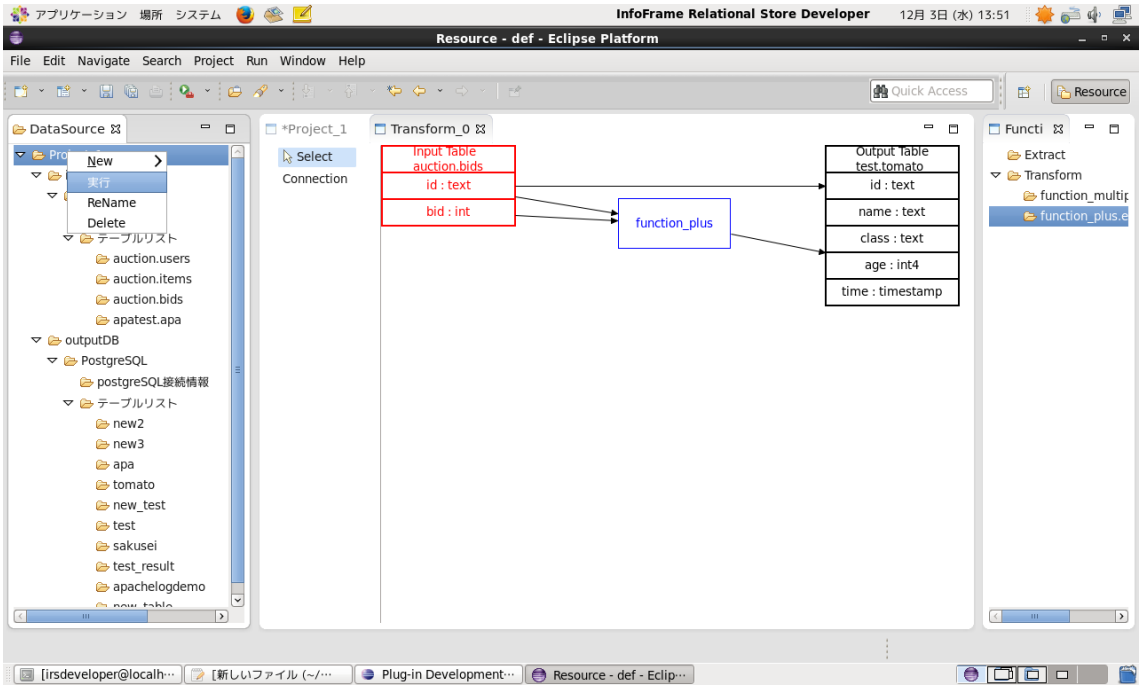
   注意：Input Table のカラムを線の終点とすることができません。同様に、Output Column を線の始点とすることができません。



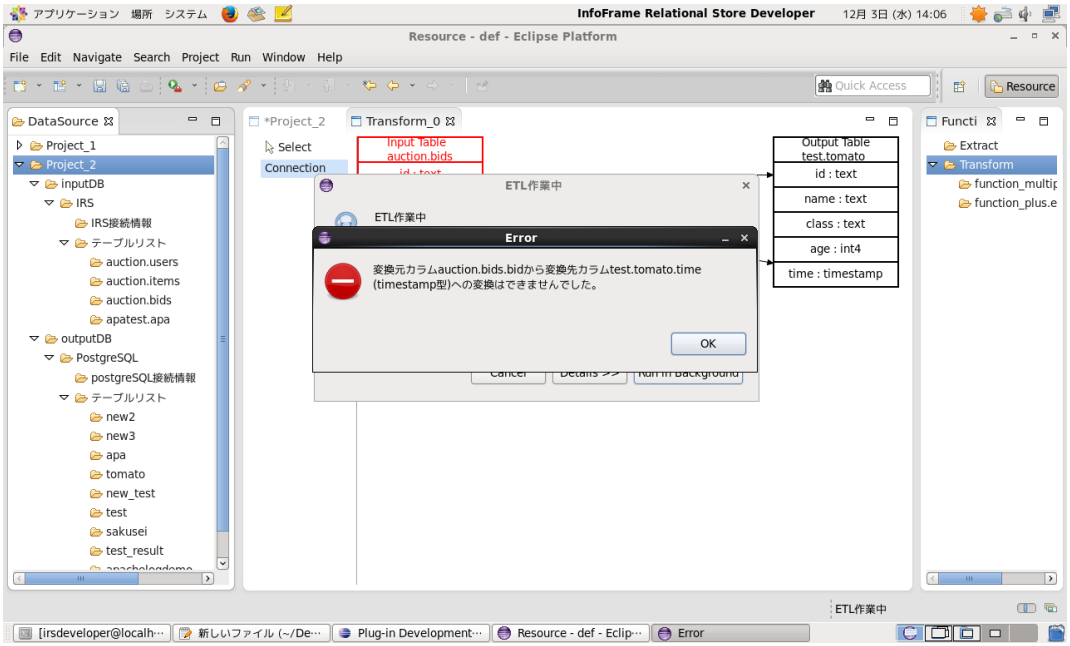
4. 左側の「Select」を選び、テーブルを選択してドラグすることで、テーブルの位置が調整できます。
5. 選択した線を右クリックして、「Delete」を選択することで、線の削除ができます。
6. Transform 編集画面を右クリックして、「Undo」を選択することで、実行済みの操作が取り消せます。
7. Transform 編集画面を右クリックして、「Redo」を選択することで、取り消された操作がやり直せます。

11. プラグインを実行する

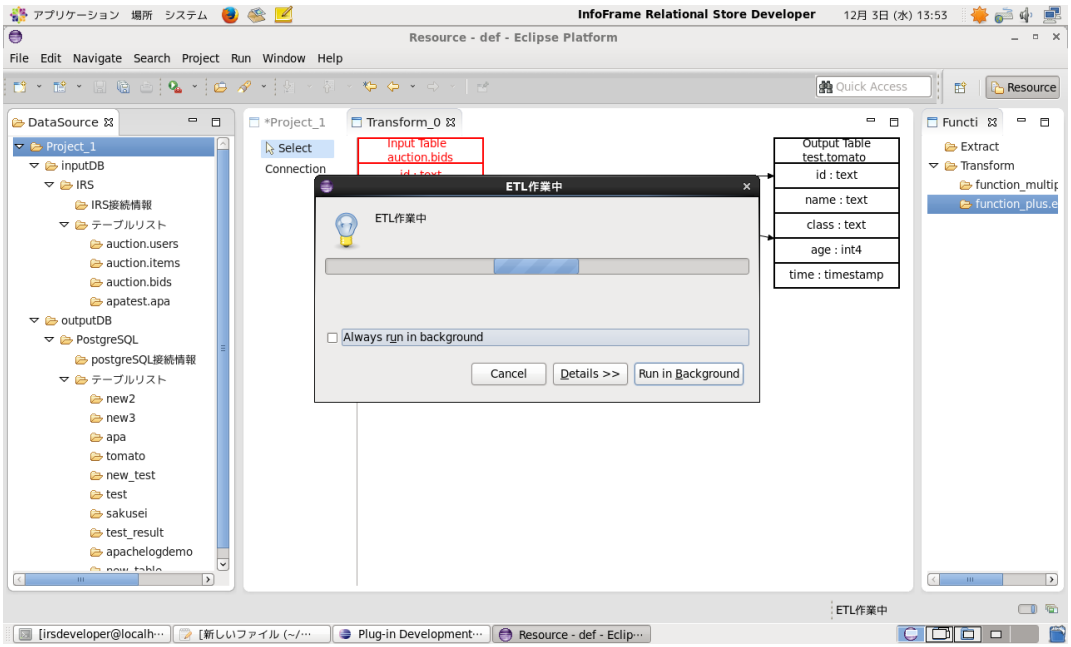
1. Extract 条件編集がすべて終わったら、プロジェクトの実行ができます。プロジェクト名を右クリックして、「実行」を選択すると、プロジェクトが実行されます。



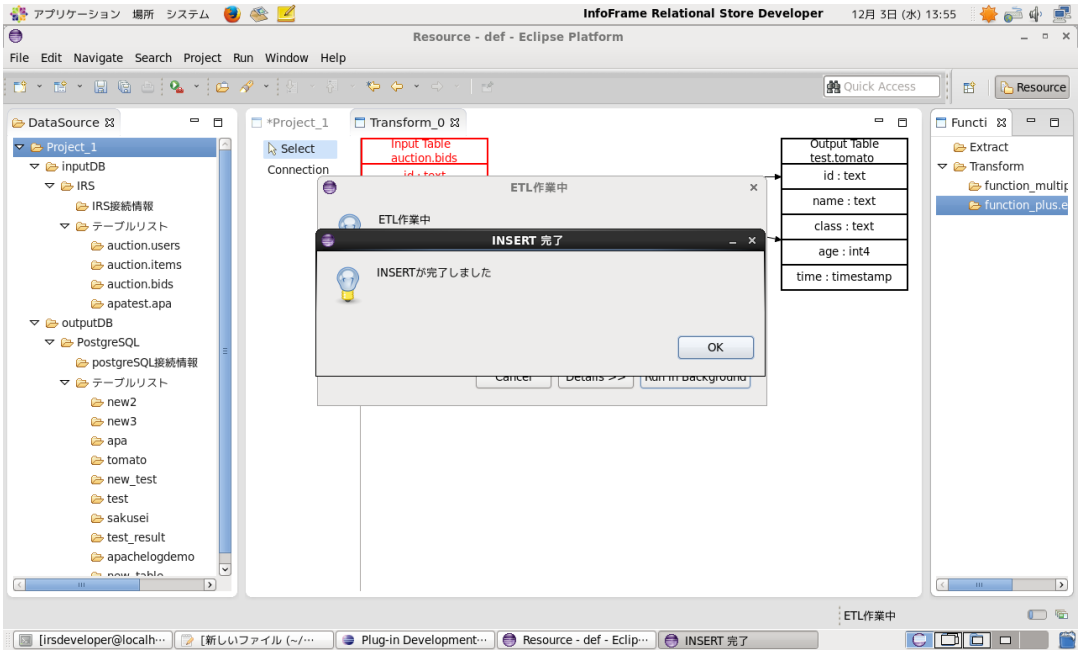
2. プロジェクトの実行が失敗した場合、エラーメッセージが表示されます。



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。



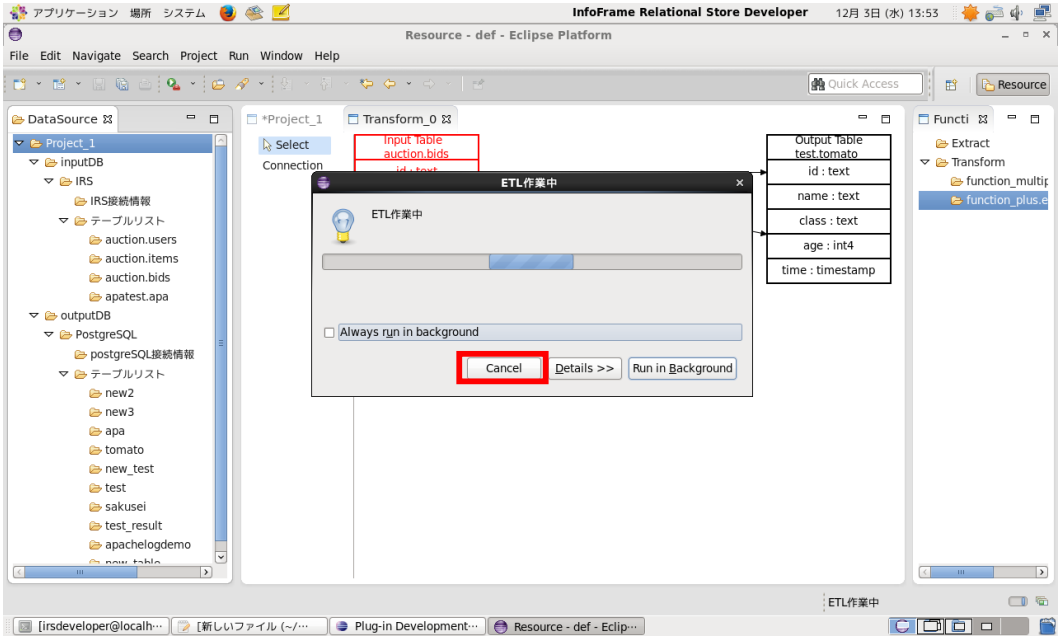
4. 作業が完了すると以下のように完了した旨の表示がされます。OK ボタンを押すと作業中ダイアログと共に完了ダイアログが消えます



12. ETL 作業を中断する

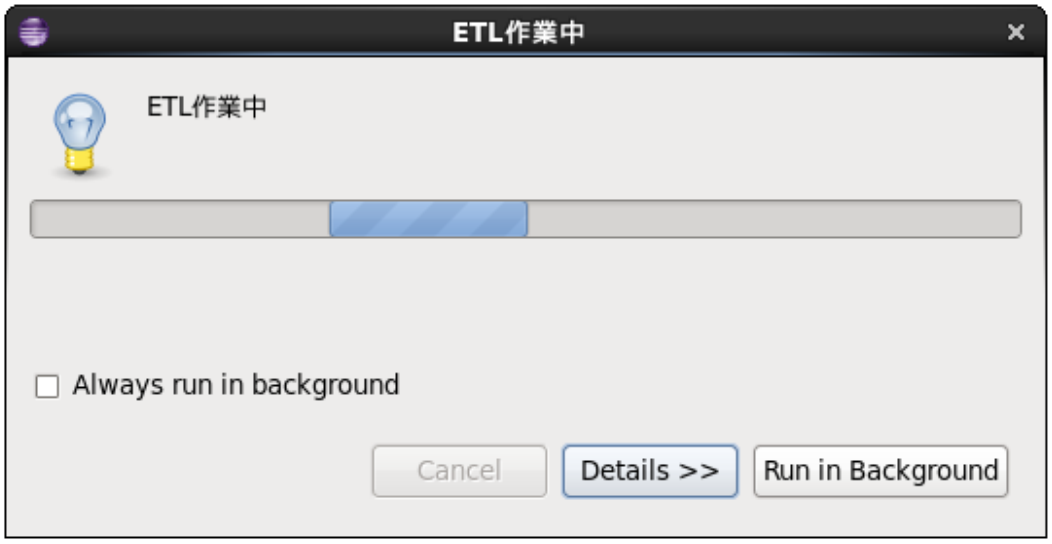
ETL 作業は中断することができる

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。

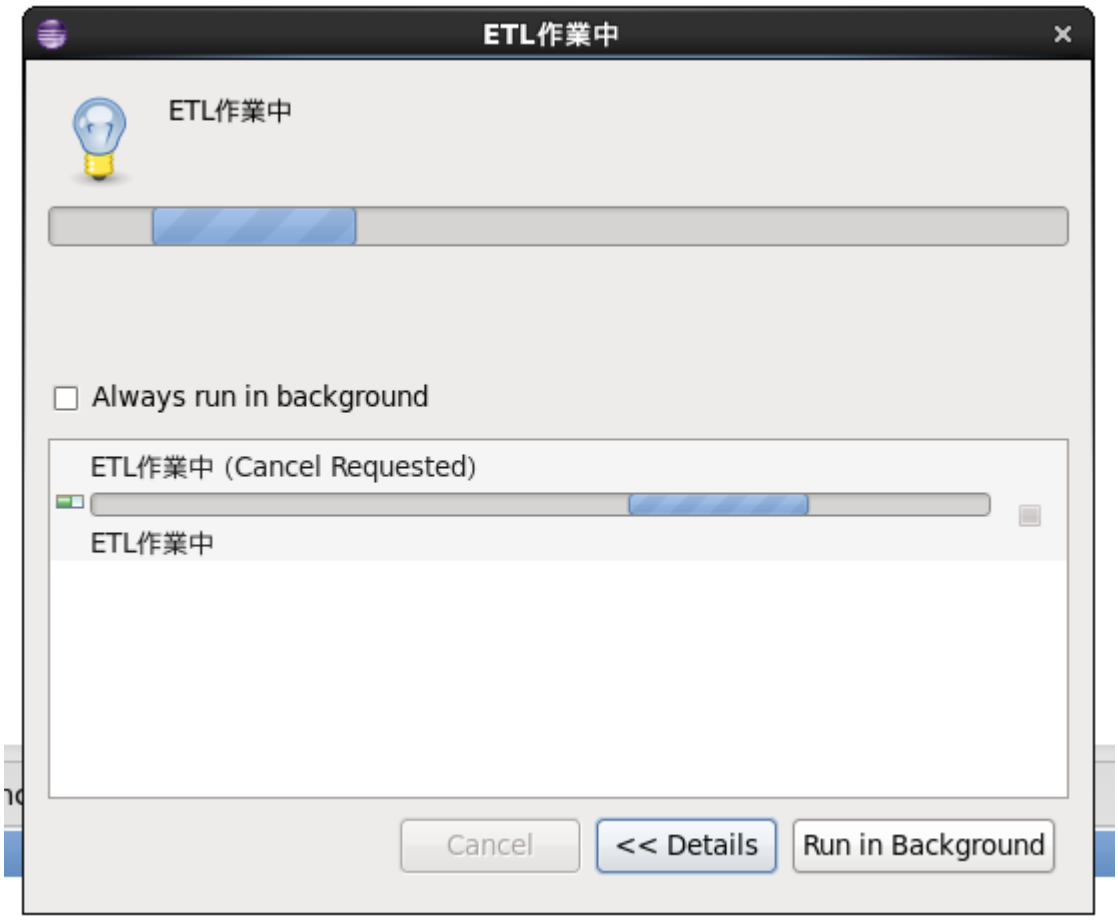


これ移行はダイアログを拡大した画像用いて説明を行います。

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。

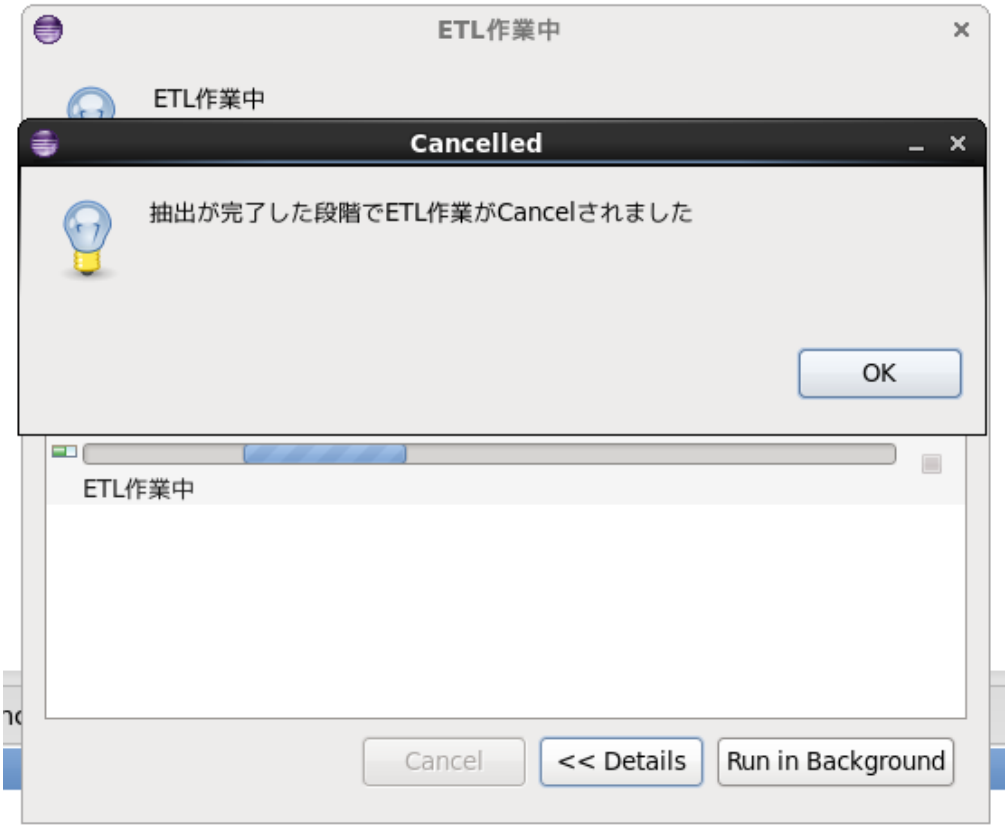


3. キャンセルボタンを押した後に、Detail ボタンを押すと以下のように ETL 作業が中断準備中であることが表示されます。

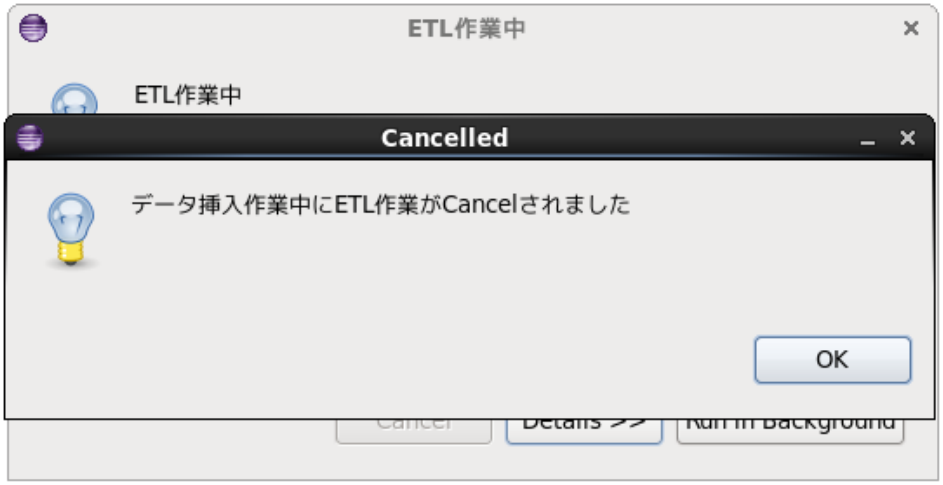


4.　中断するタイミングは２つあります。

(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。

第 1 節　　セットアップマニュアル

本節では、本ツールのプラグインを Eclipse に導入する手順を示したセットアップマニュアルの内容を記述する。納品フォルダ内のマニュアル類フォルダにセットアップマニュアルの内容を示したファイルを配置した。

説明には

Eclipse

Eclipse for RCP and RAP Developers

Version: Kepler Service Release 2

Build id: 20140224-0627

http://www.eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/keplersr2　からダウンロードしたものを使用した。

手順を以下に示す。

1. Eclipse の plugins フォルダにプラグインファイルをコピーする。

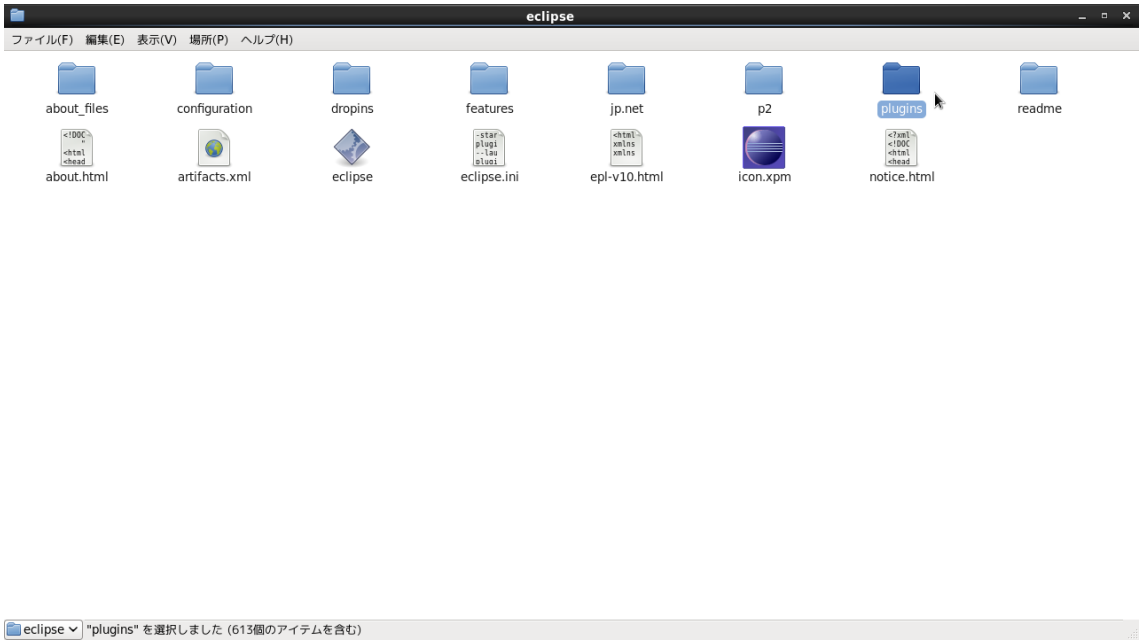


図 72

2. Eclipse を起動する

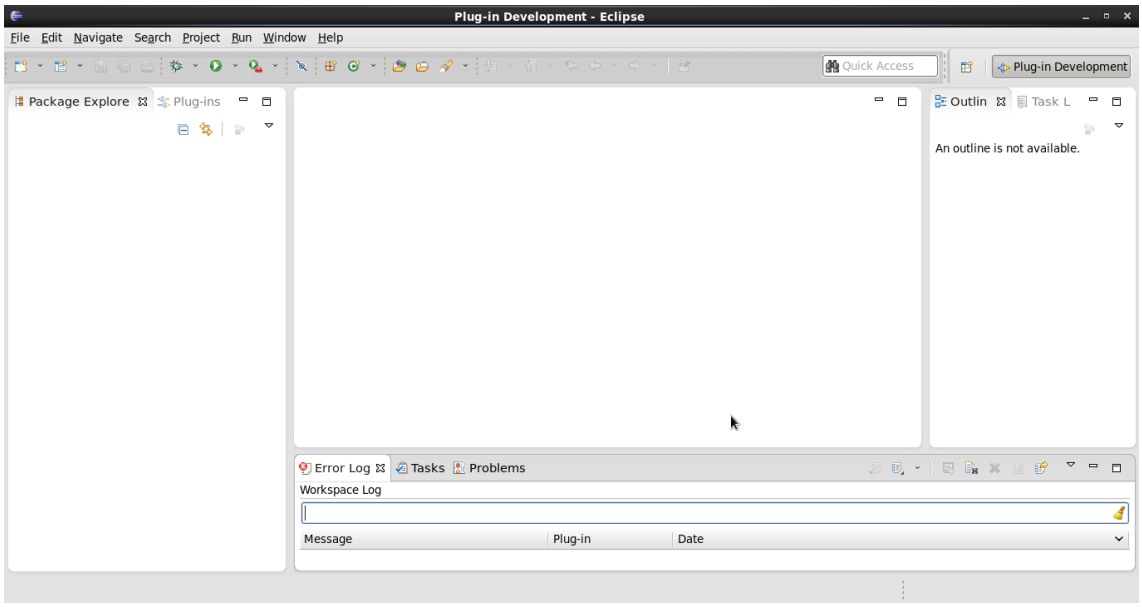


図 73

3. 全てのビューを閉じる

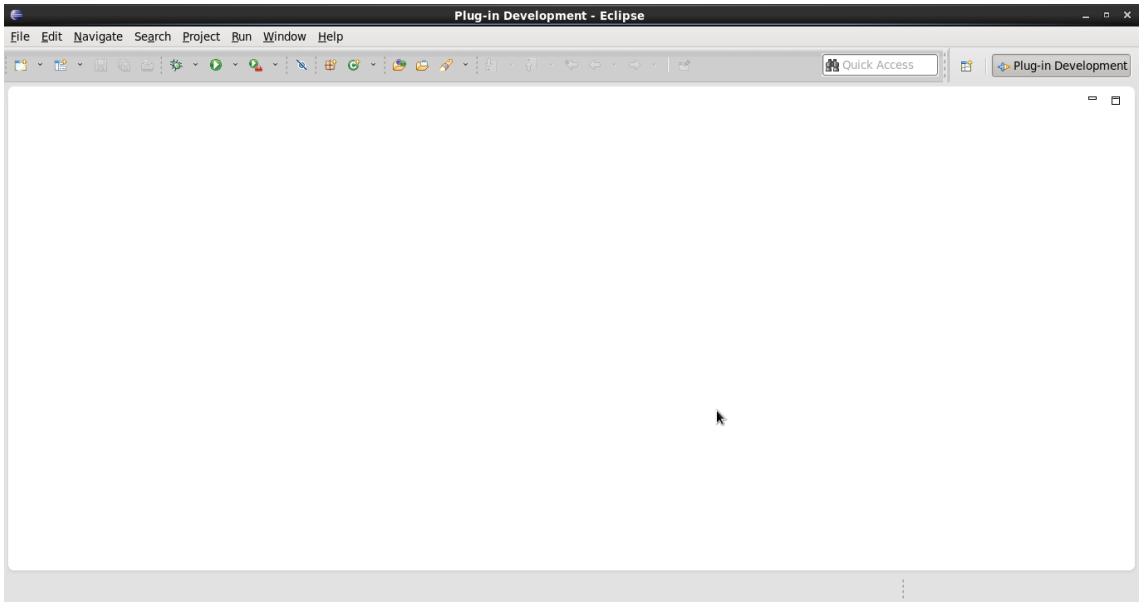


図 74

4. ETL 機能に必要なビューを表示するために、window->Show View->Other…をクリックする。

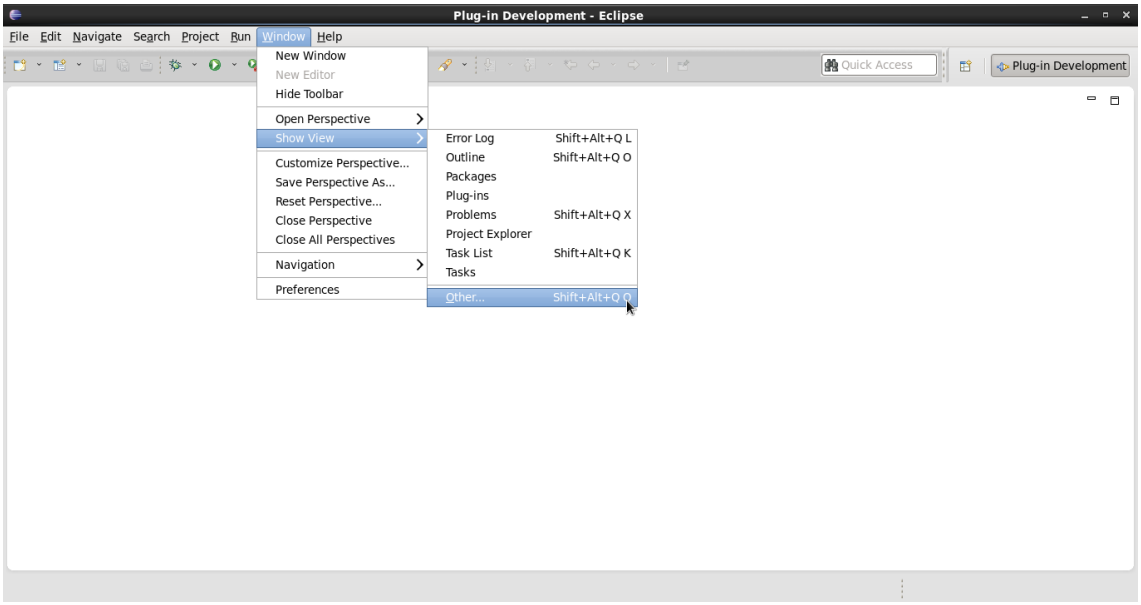


図 75

5. Show View ウィンドウで、Other 中の DataSource をダブルクリックする。

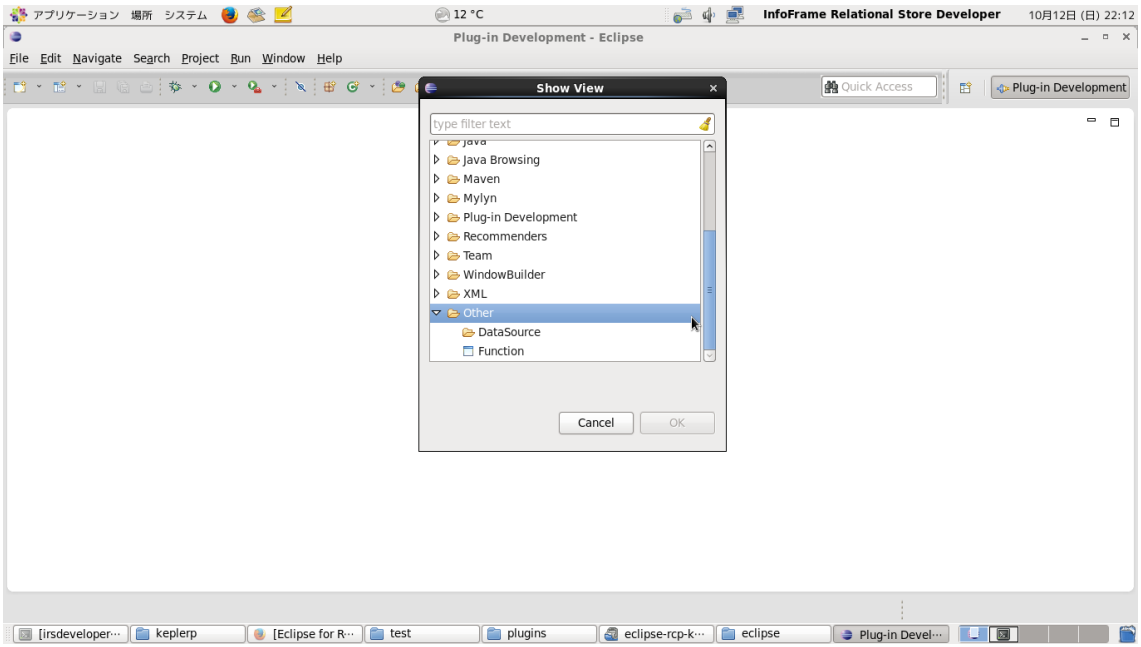


図 76

6. DataSource ビューが画面に追加される。

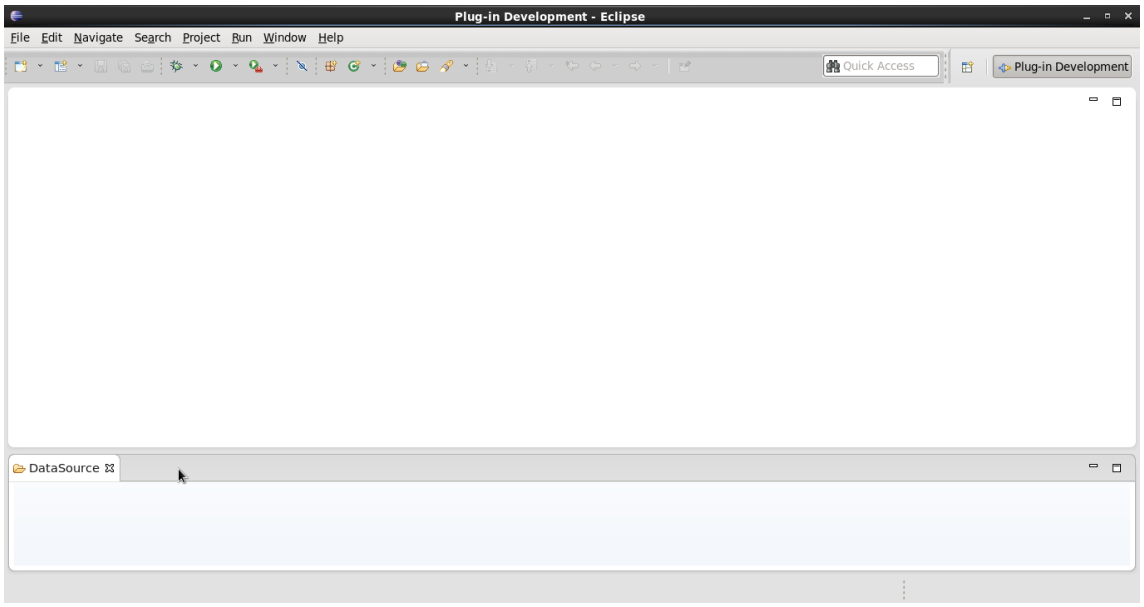


図 77

7. 手順 5 と同様に Function ビューも追加する。

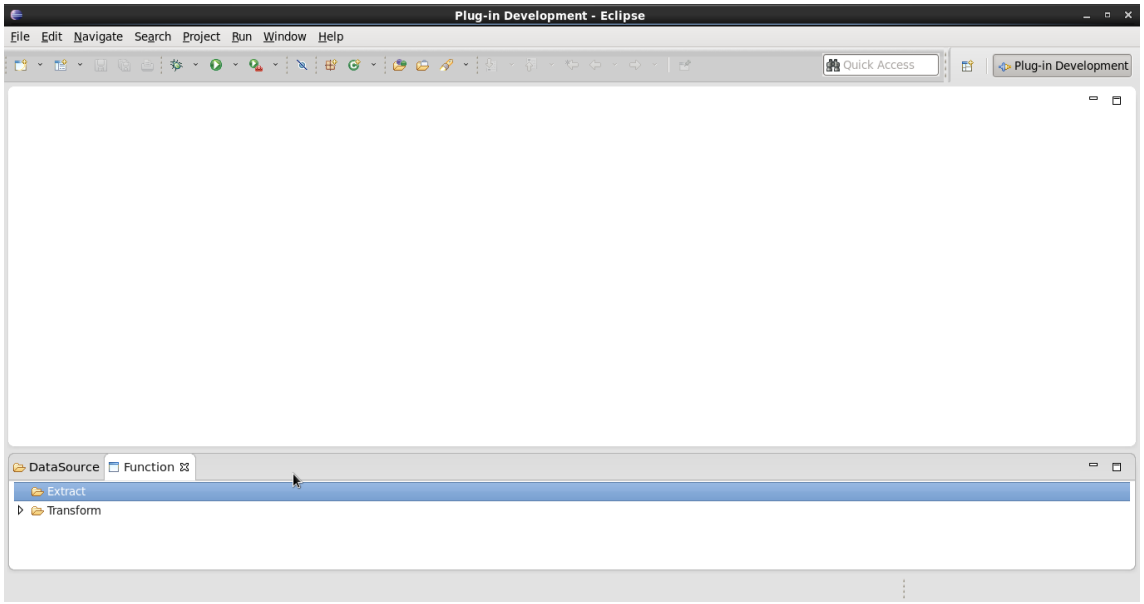


図 78

8. 操作しやすいようにビューの位置を調整する。左側に DataSource ビュー、右側に Function ビューが基本位置となる。

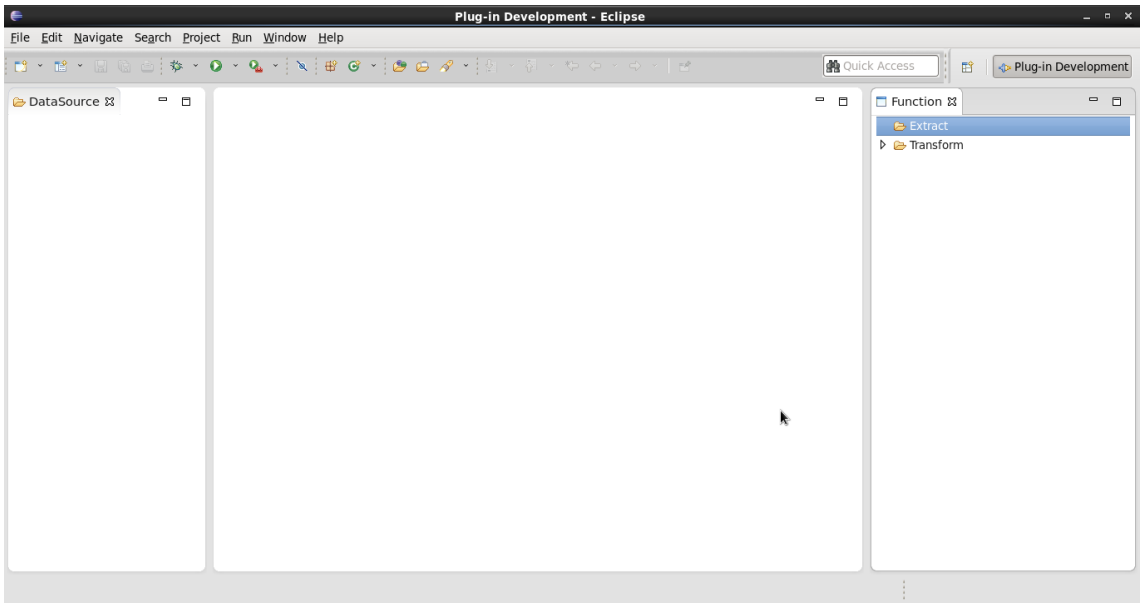


図 79

以上でプラグインの導入は終了である。

第２節　Transform プラグイン作成マニュアル

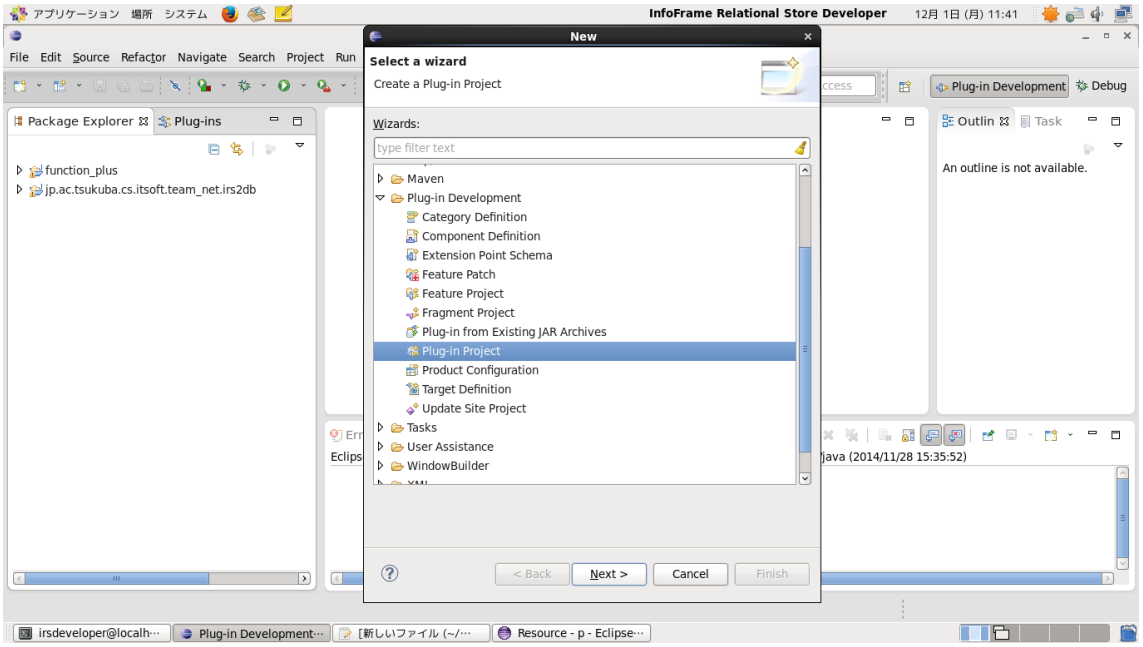
ETL ツールにとって、多様な Transform 機能に対応していることは必須事項である。NET ツールでは Transform 機能をプラグイン化することで、NET ツールの変更なしに、ユーザが作成した Transform 機能を追加することができる。

本マニュアルで使った Eclipse のバージョン情報を以下に示す。

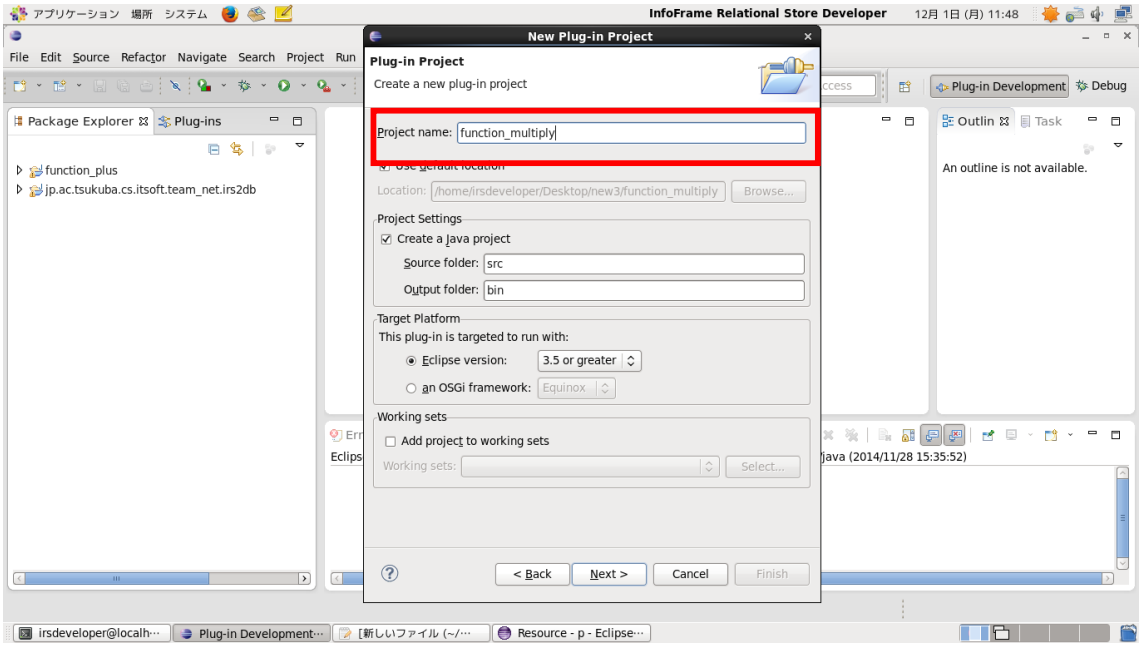
Eclipse for RCP and RAP Developers	
Version	Kepler Service Release 2
Build id	20140224-0627

以下は Transform プラグインの作成手順である。

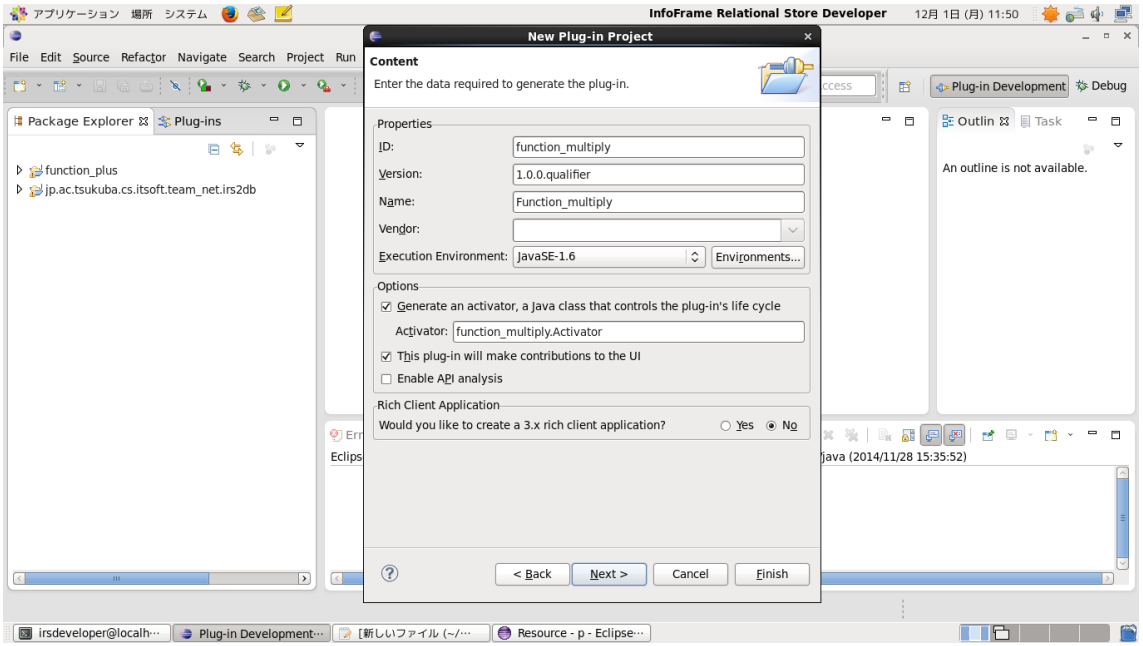
1. File → New → Other → Plug-in Development の Plug-in Project を選択 → Next



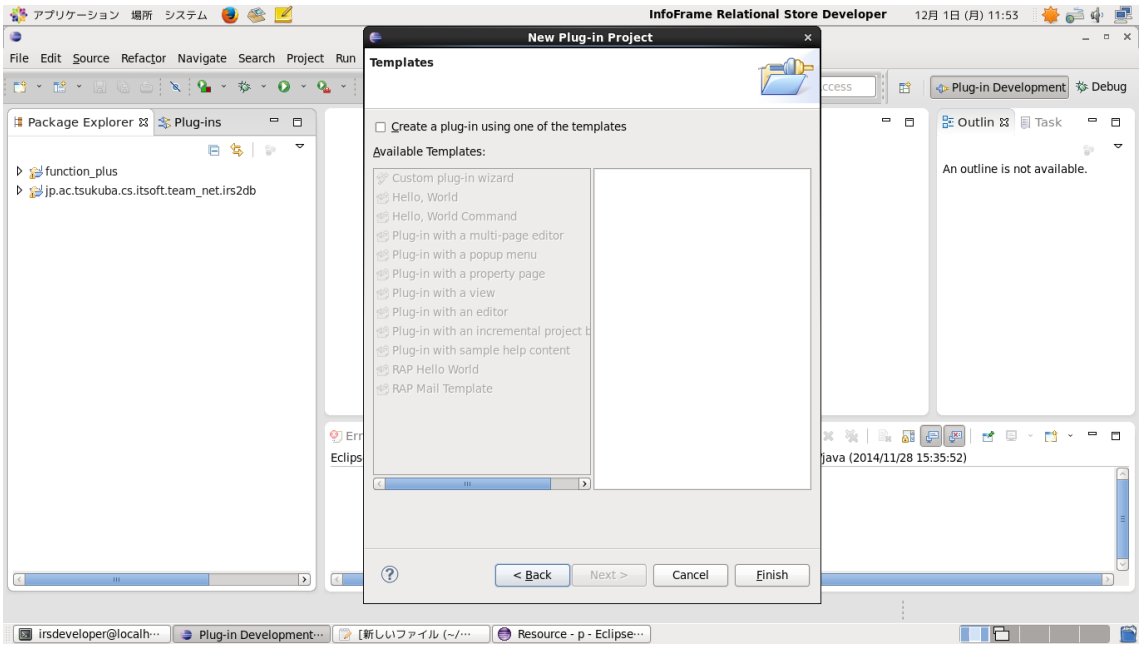
2. プラグインの名前を入力 → Next



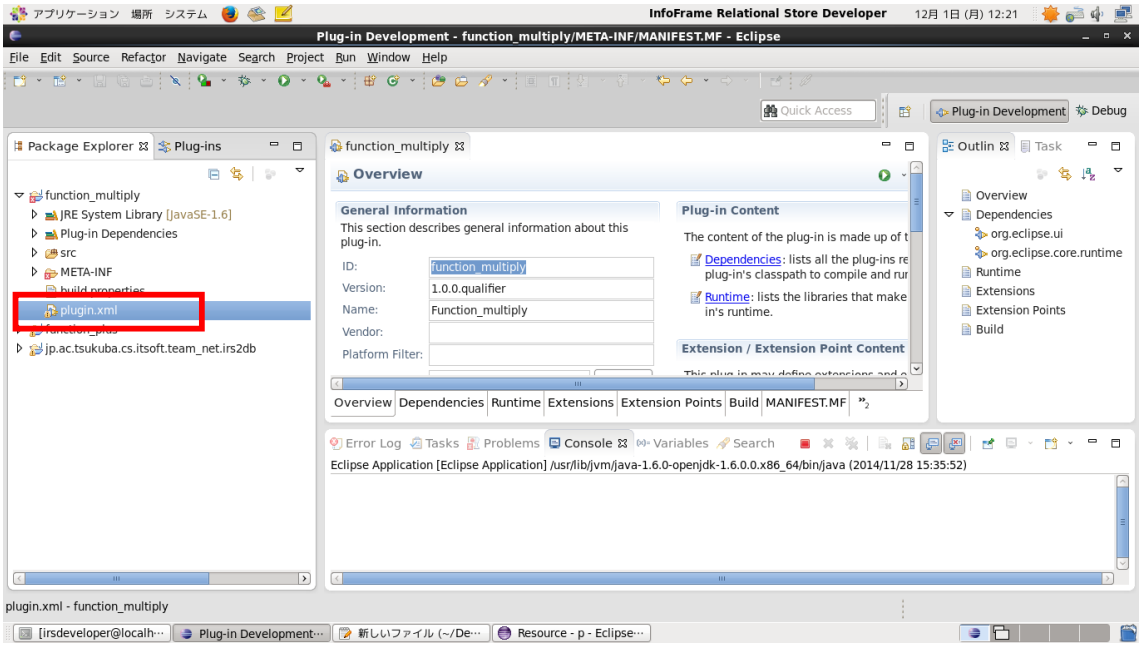
3. 指定がなければ Next



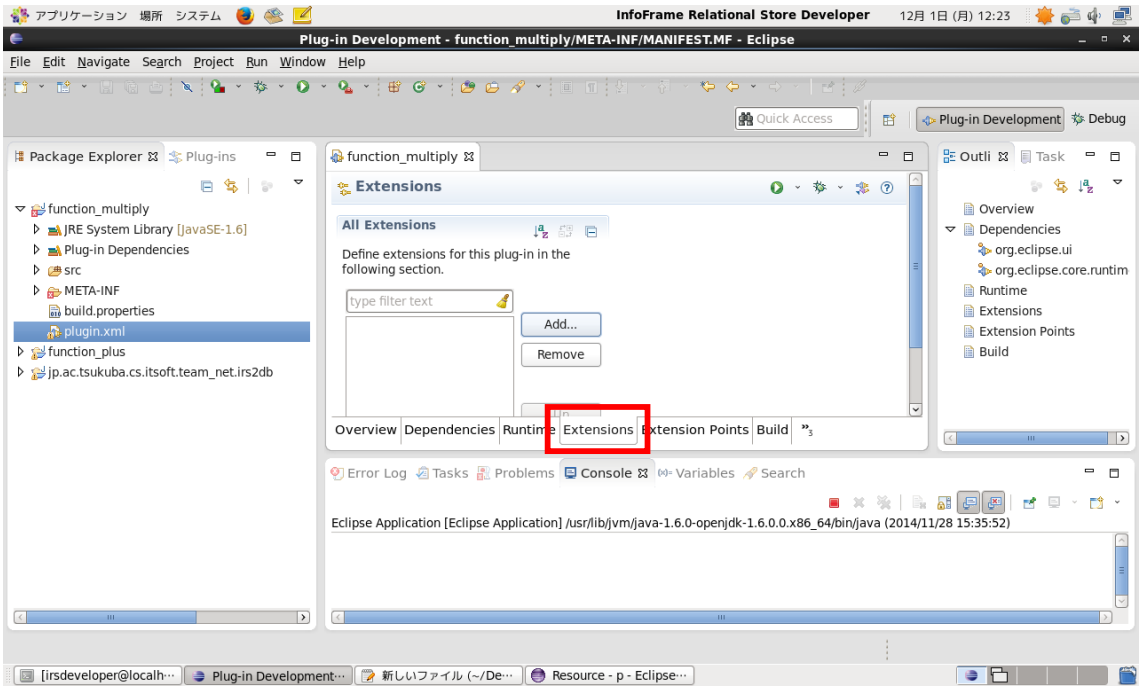
4. 「Create a plug-in using one of the templates」のチェックを外し、Finish ボタンを押すと、作成するプラグインのプロジェクトが作成される



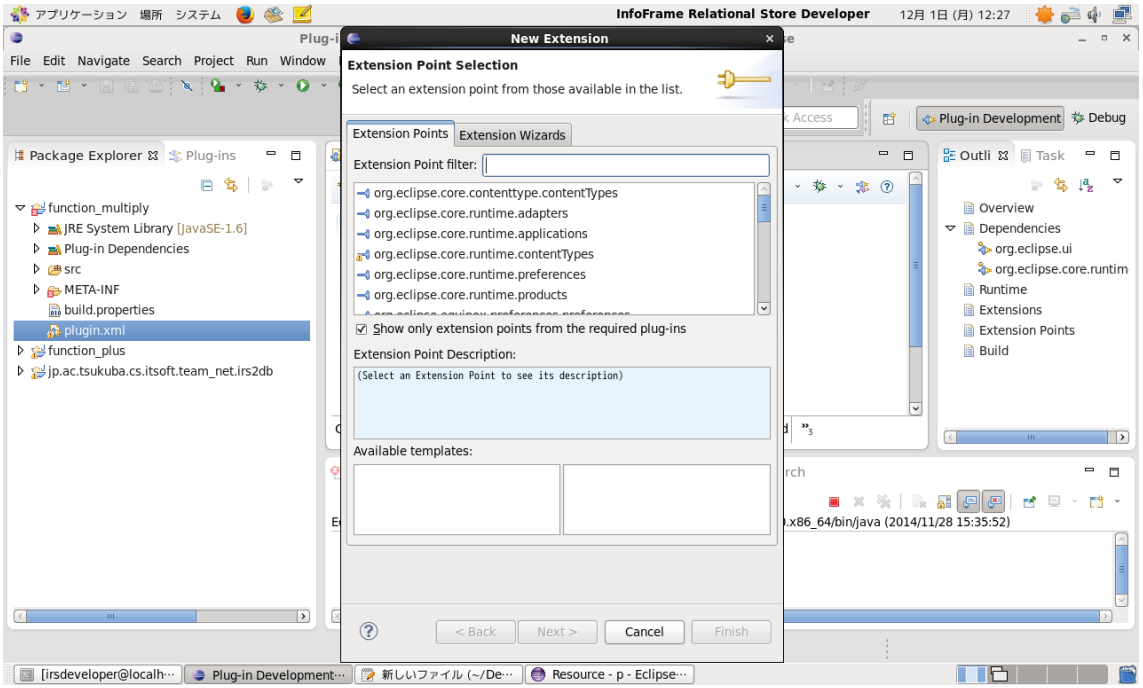
5. 作成するプラグインのプロジェクトを展開し、plugin.xml をダブルクリックして、プラグインの設定ファイルを開く



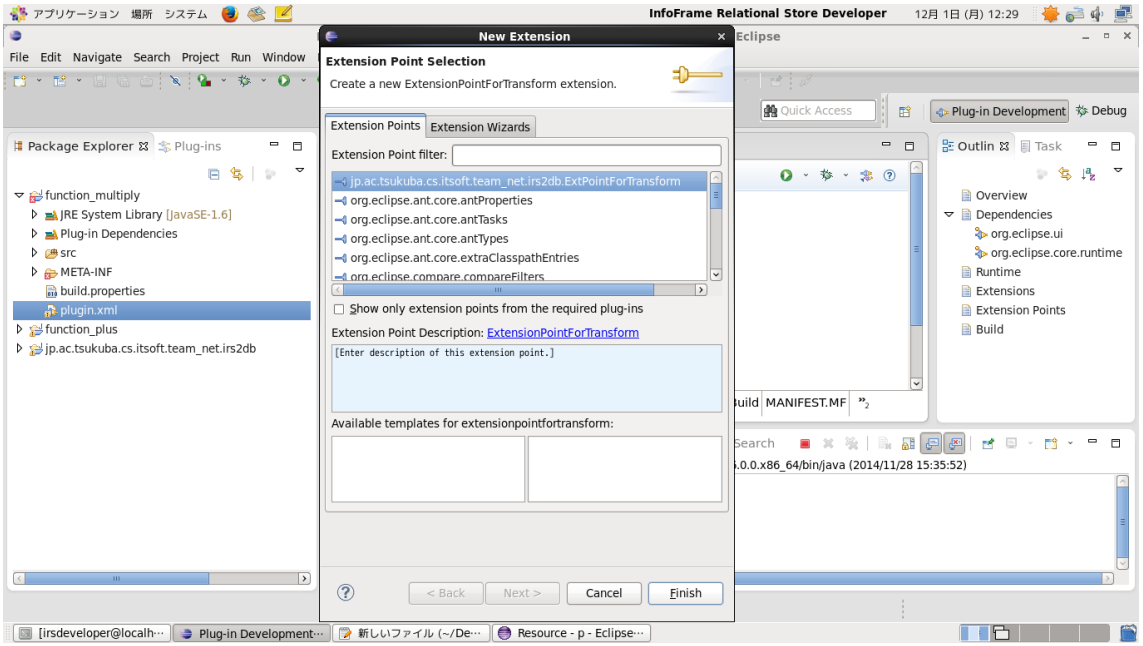
6. 「Extensions」タグを選択する



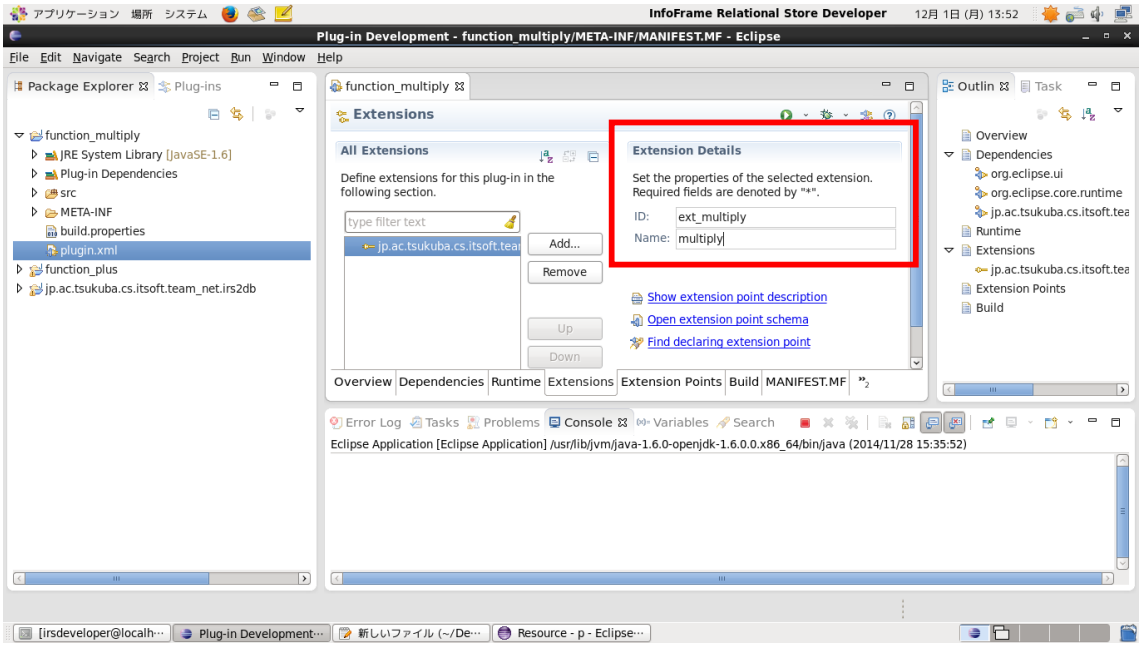
7. 「Add」ボタンをクリックして、Extension Points 選択画面を開く



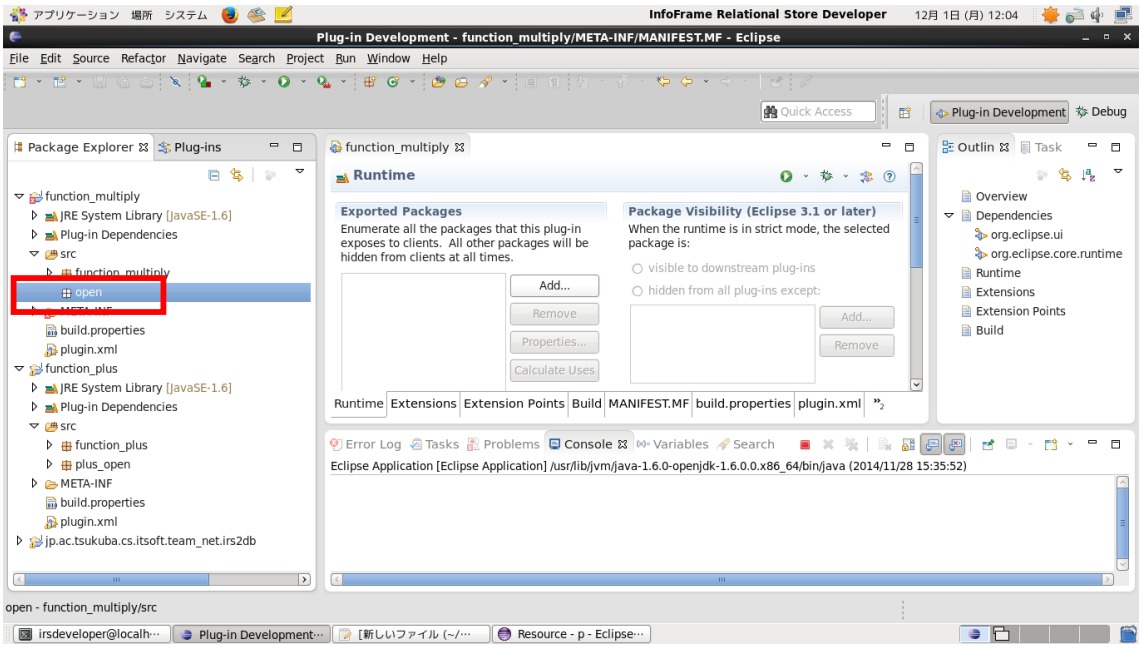
8. 「Show only extension points from the required plug-ins」のチェックを外し、「jp.ac.tsukuba.cs.itsoft.team_net.irs2db.ExtPointForTransform」を選んで、「Finish」ボタンを押す



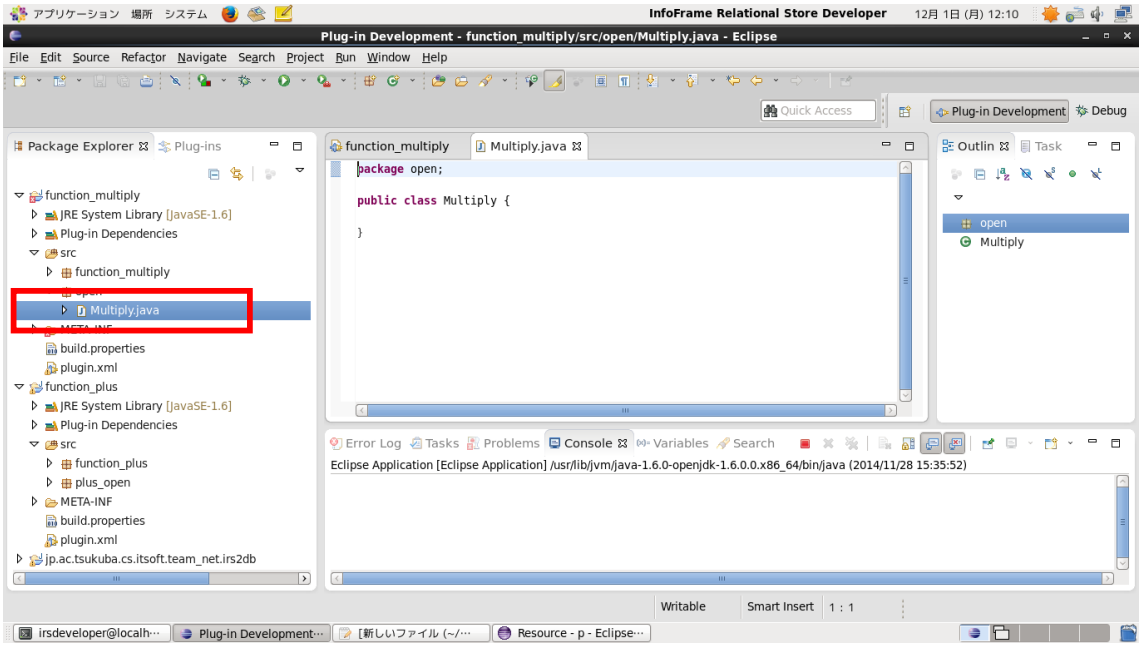
9. 拡張の ID と Name を入力して保存する



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
```

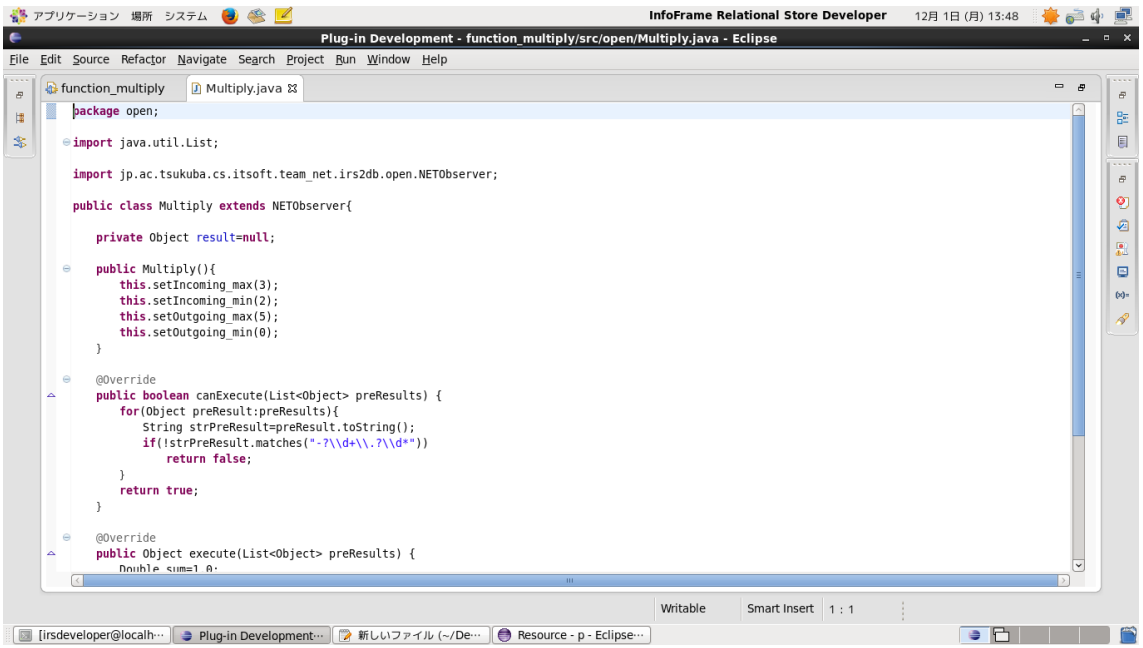
```
        if(!strPreResult.matches("-?¥¥d+¥¥.?¥¥d*")) //正規表現で数字であるかどうかチェック
          return false;
      }
      return true;
    }
```

（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;
}
```

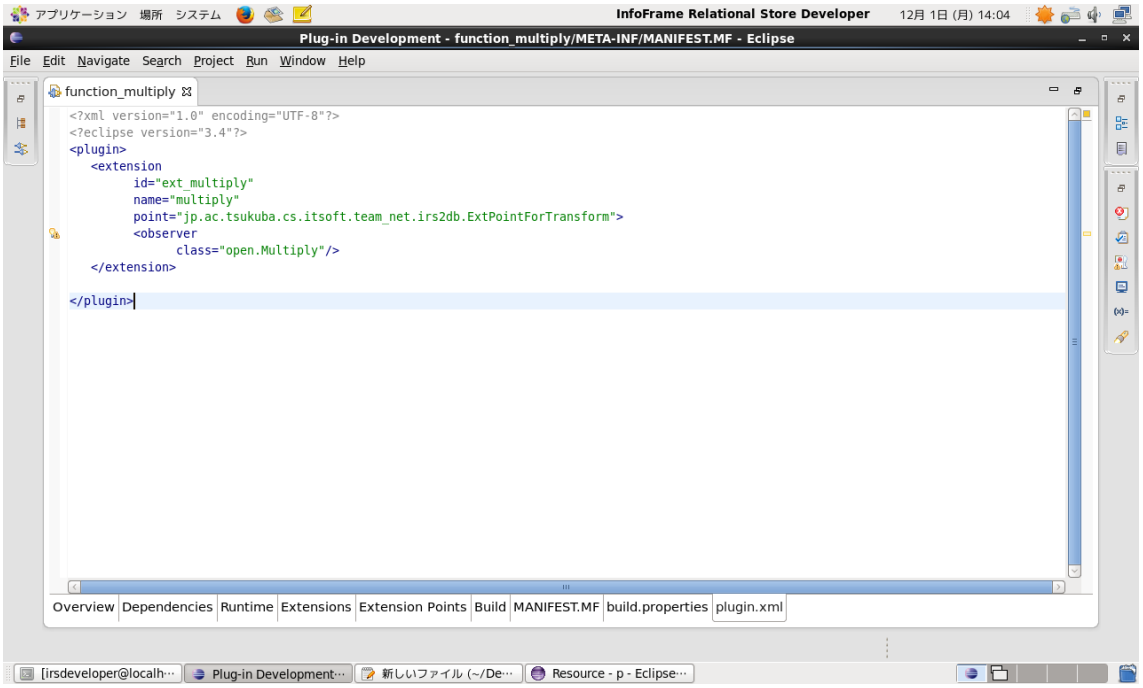


備考：

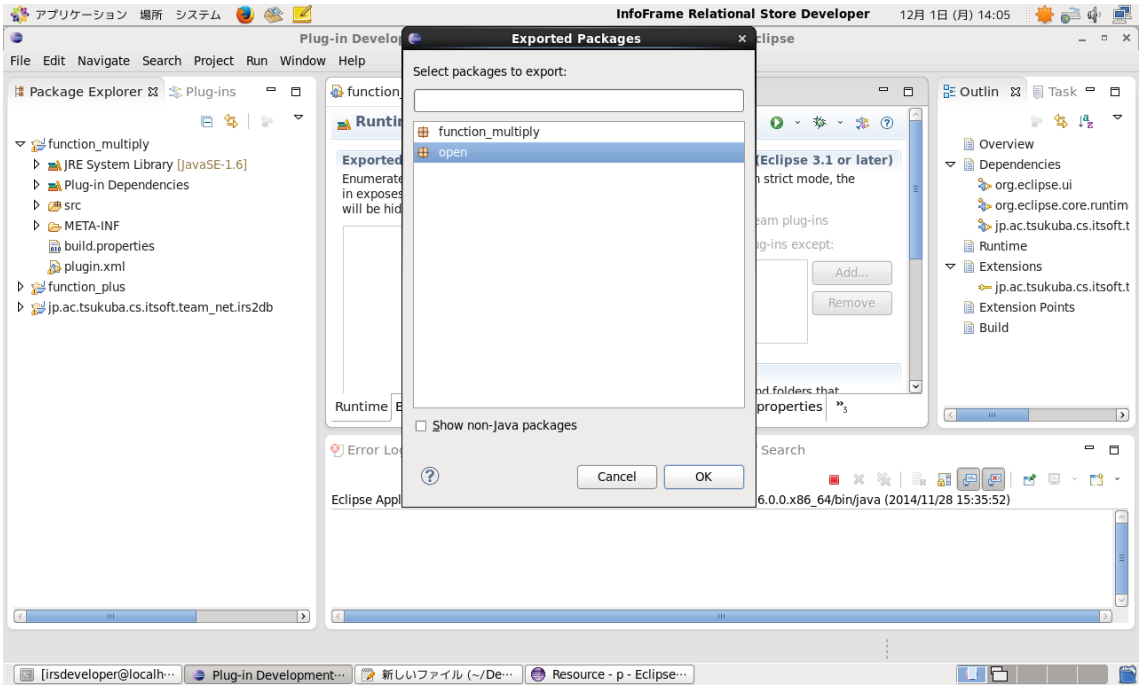
NET ツールが Transform プラグインを実行する時に、まず canExecute メソッドでプラグ

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。

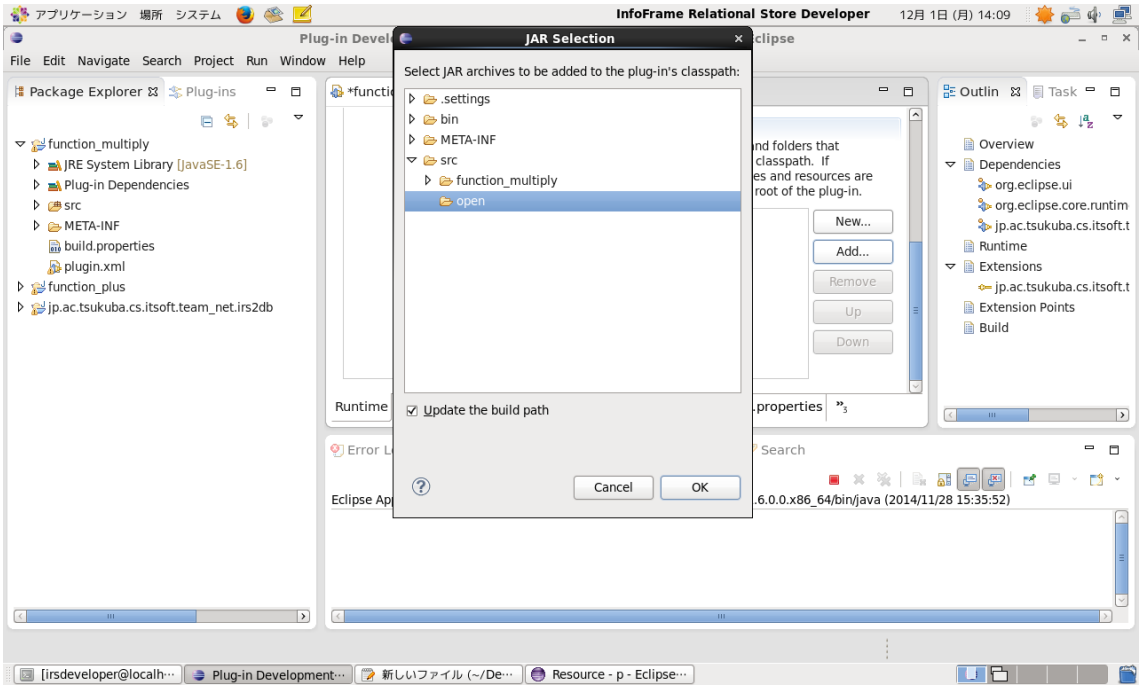
13. plugin.xml を開き、<extension>の中に<observer/>タグを作成して、class 属性の値として 11 に作成したクラスを指定する
    例：
    <observer class="open.Multiply"/>



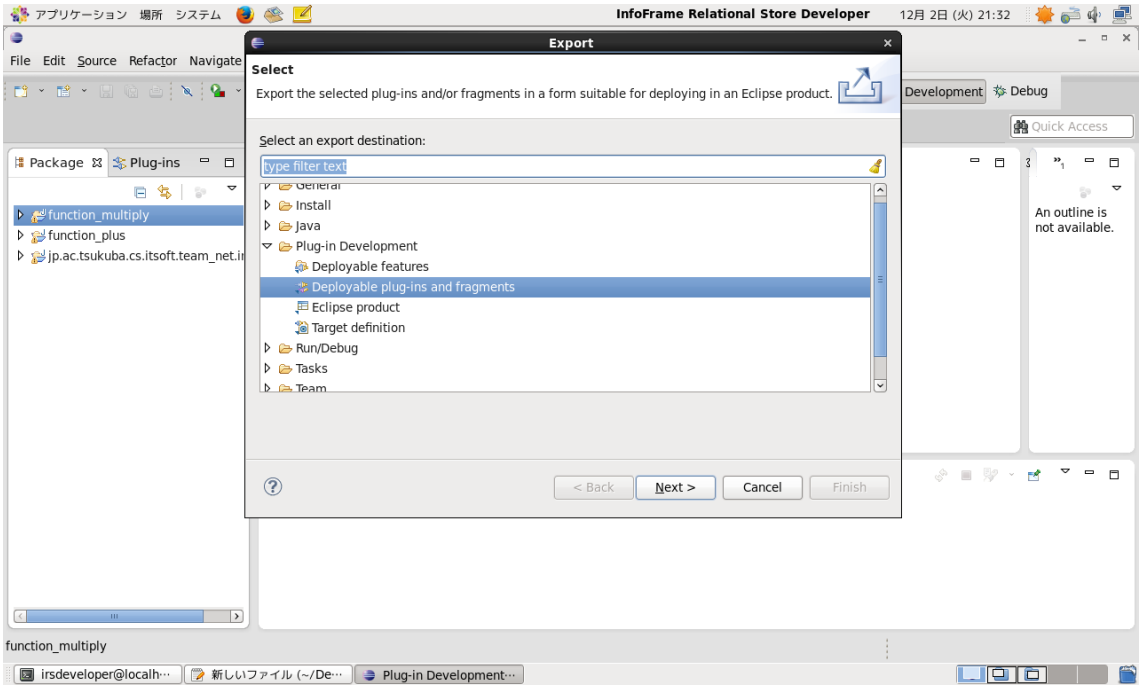
14. Runtime を選択し、Exported Packages エリアで「Add」ボタンを押して、10 で作成したパッケージを選び、「OK」ボタンを押す



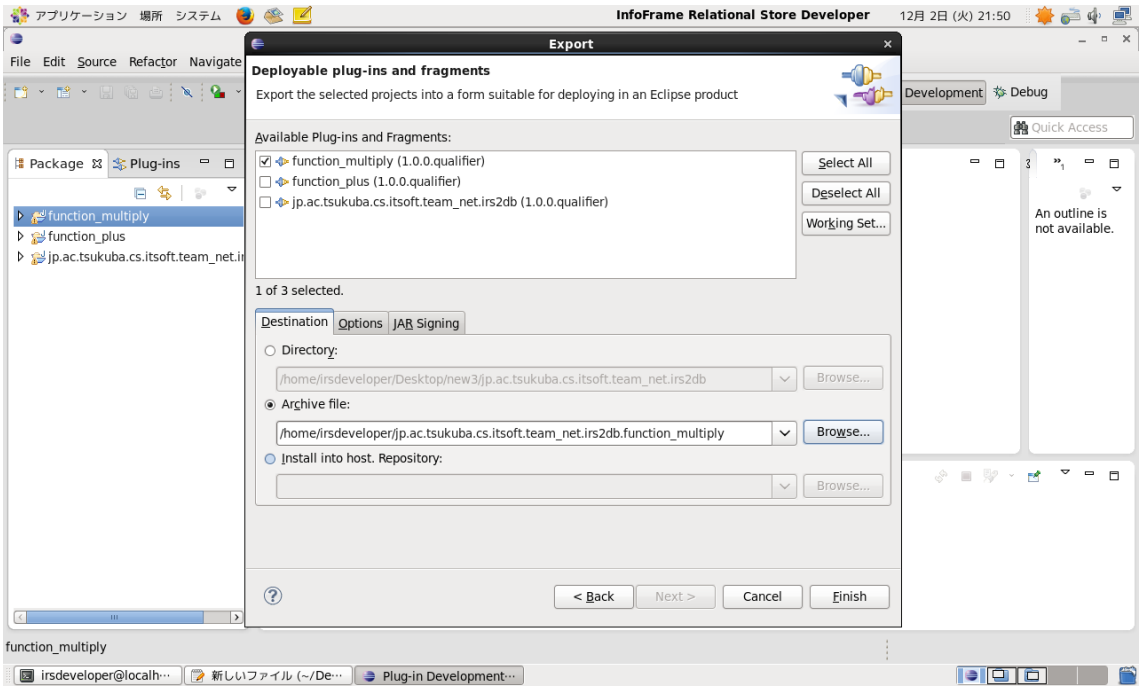
15. Classpath エリアで「Add」ボタンを押し、10 で作成したパッケージを選び、「OK」ボタンを押して、以上の設定を保存する



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

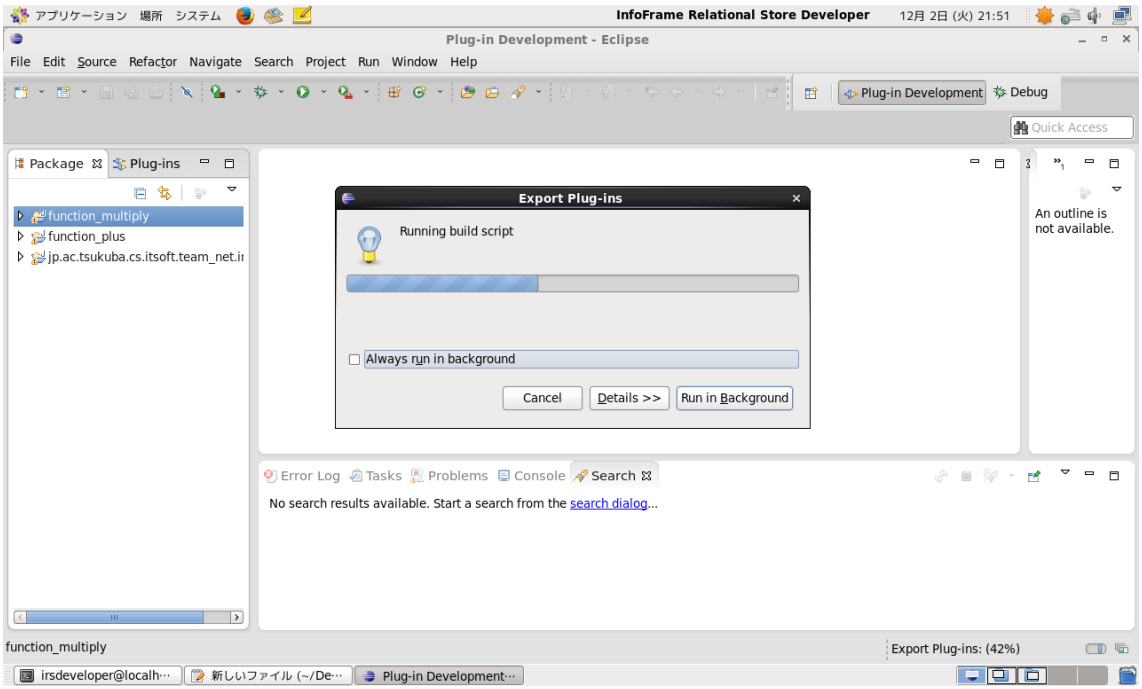


17. 上のエリアで作成するプラグインを選択し、下のエリアで「Destination」タグの「Archive File」をチェックし、エクスポートするアーカイブファイルの保存先を指定する



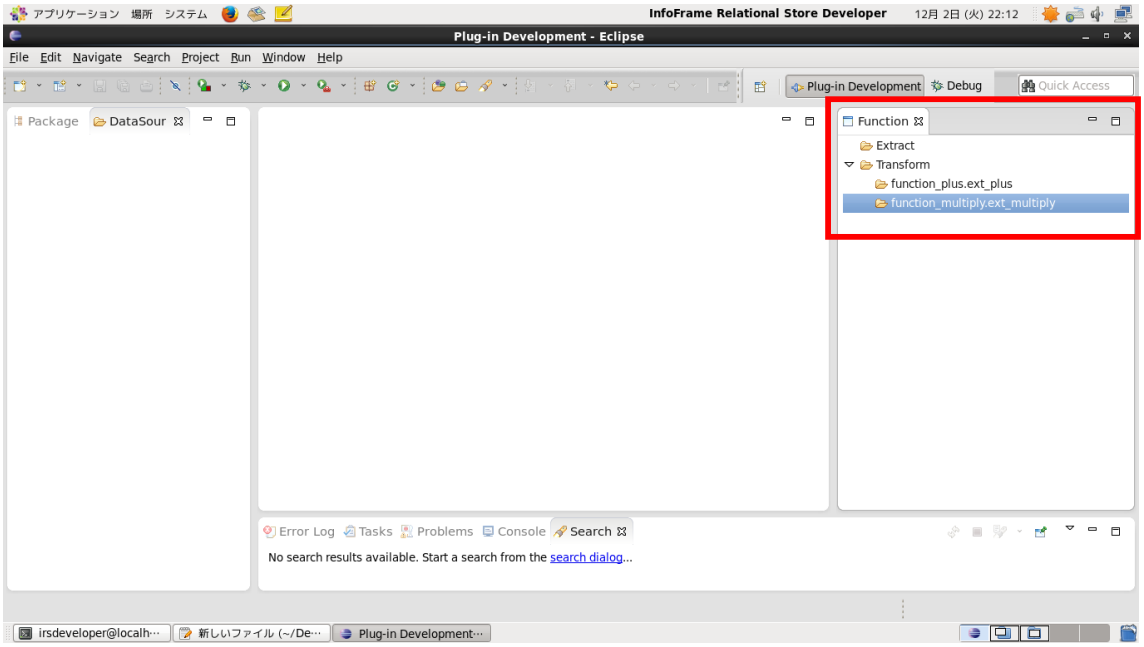
18. 「Finish」ボタンを押すと、作成するプラグインをアーカイブする zip ファイルがエク

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

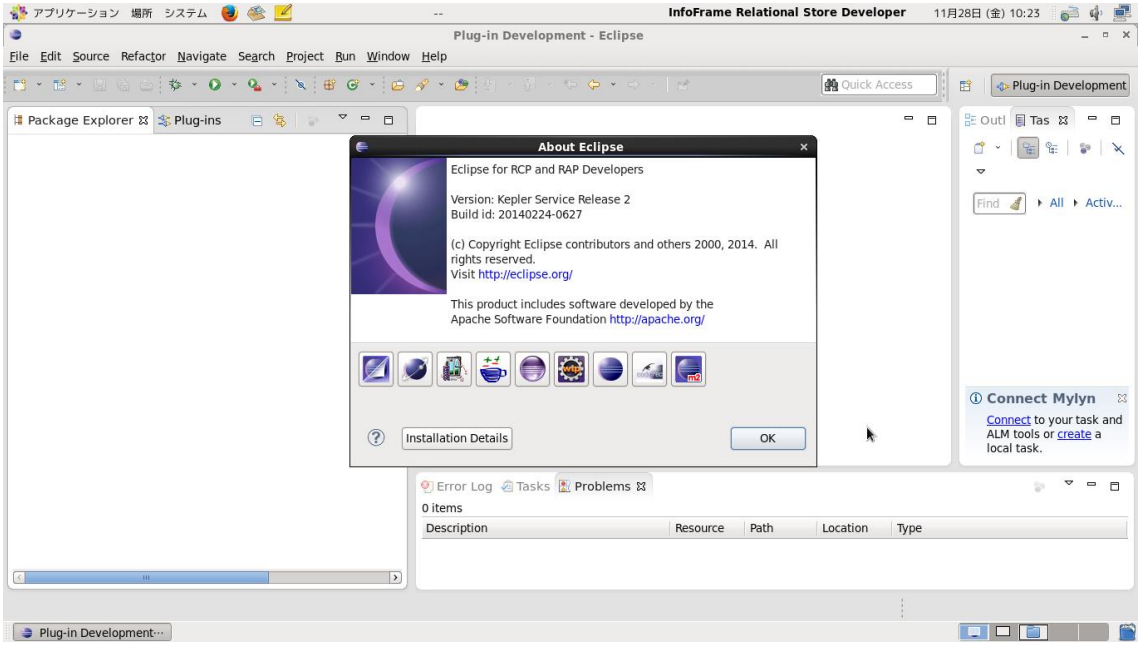


第３節　　ビルド手順書

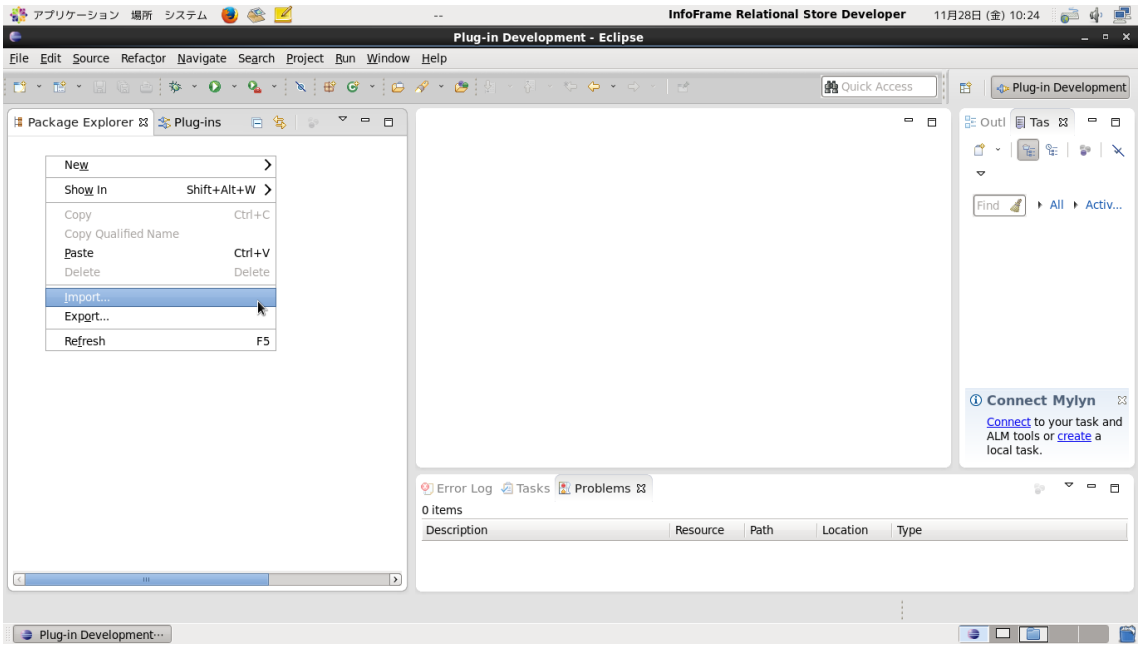
NET ツールは Eclipse を使用して開発された。Eclipse のビルド手順書を参照することで、NET ツールが使用しているパッケージの変更などの際にユーザが NET ツールを再ビルドすることができる。

納品物に含まれる Eclipse のプロジェクトの読み込みから NET ツールプラグインの jar ファイルを出力するまでの手順を説明する。

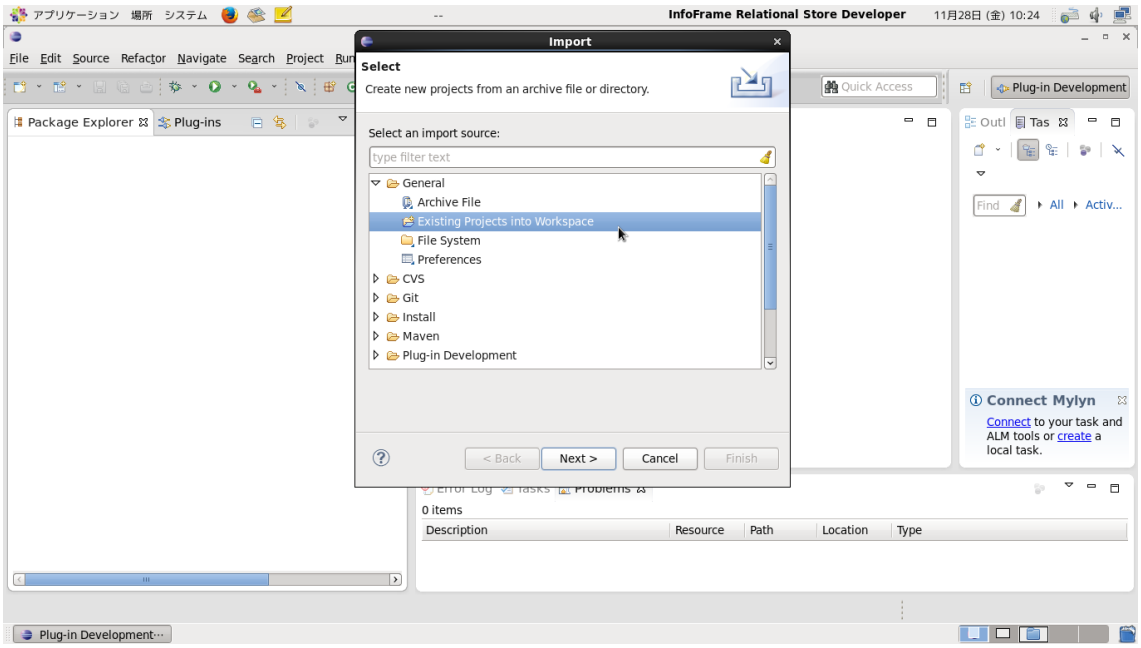
0. ビルドに使用した Eclipse のバージョンを示す。



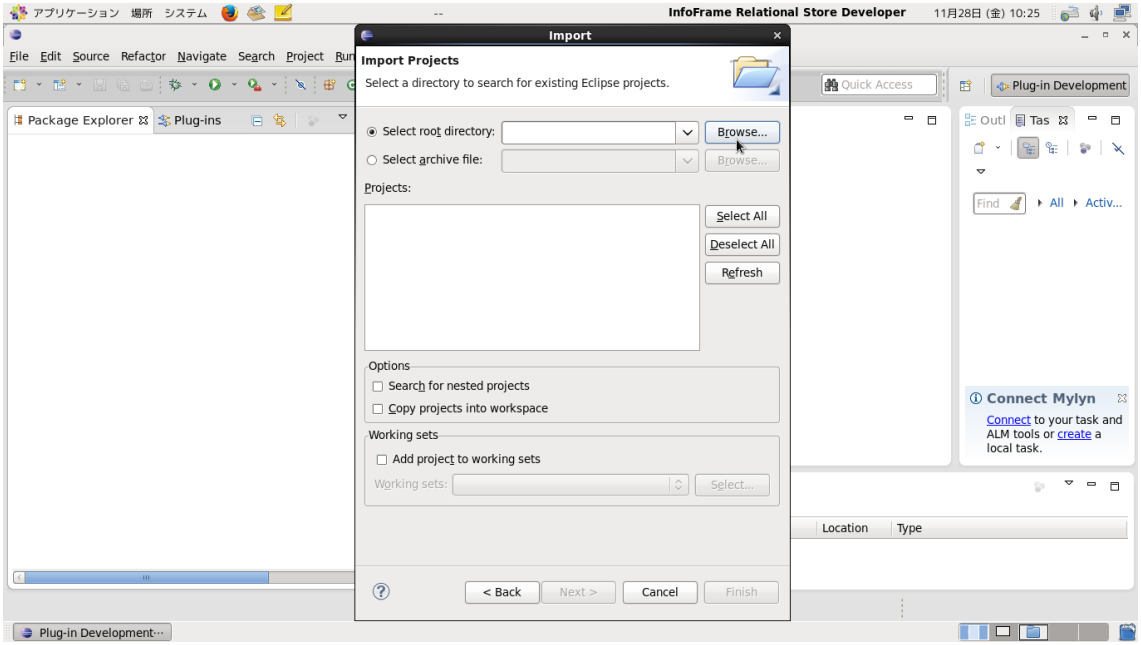
1. NET ツールのプロジェクトファイルをインポートする。



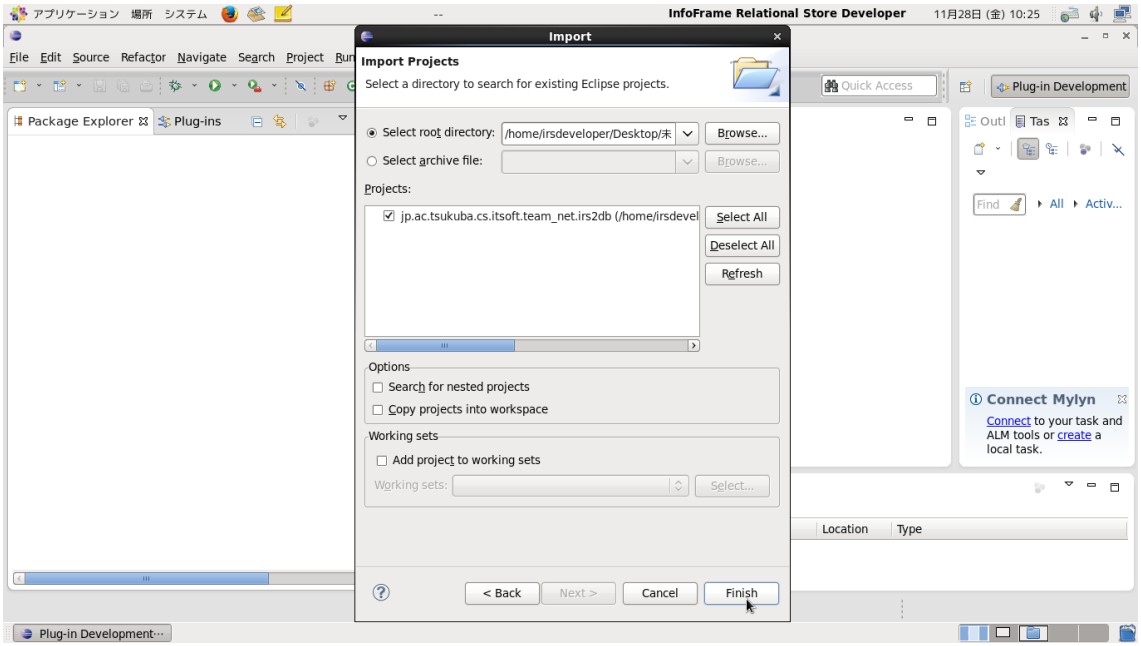
2. 「Existing Projects into Workspace」を選択する。



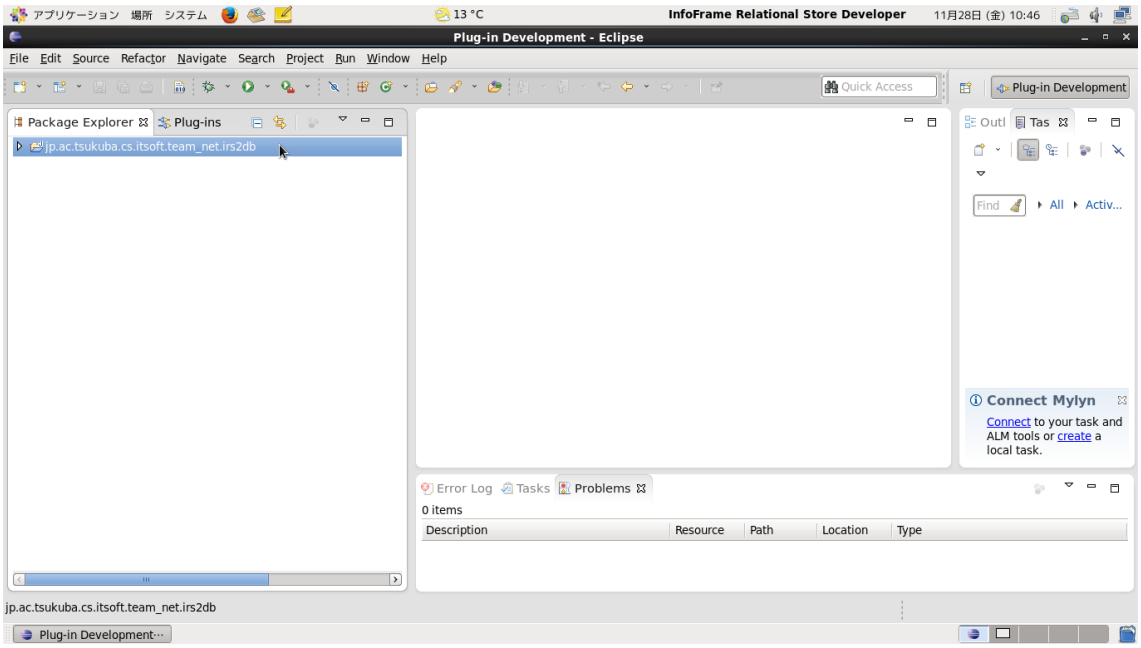
3. 「Select root directory」を選択し、NET ツールのプロジェクトを選択する。



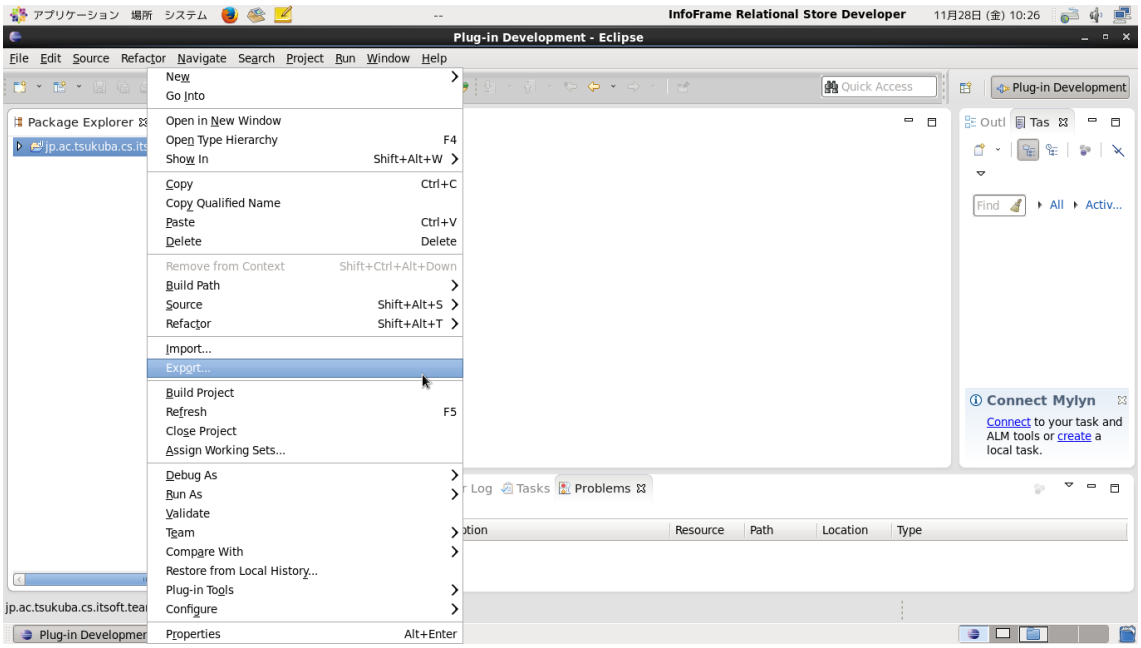
4. NET ツールのプロジェクトを選択するとこのような画面になる。



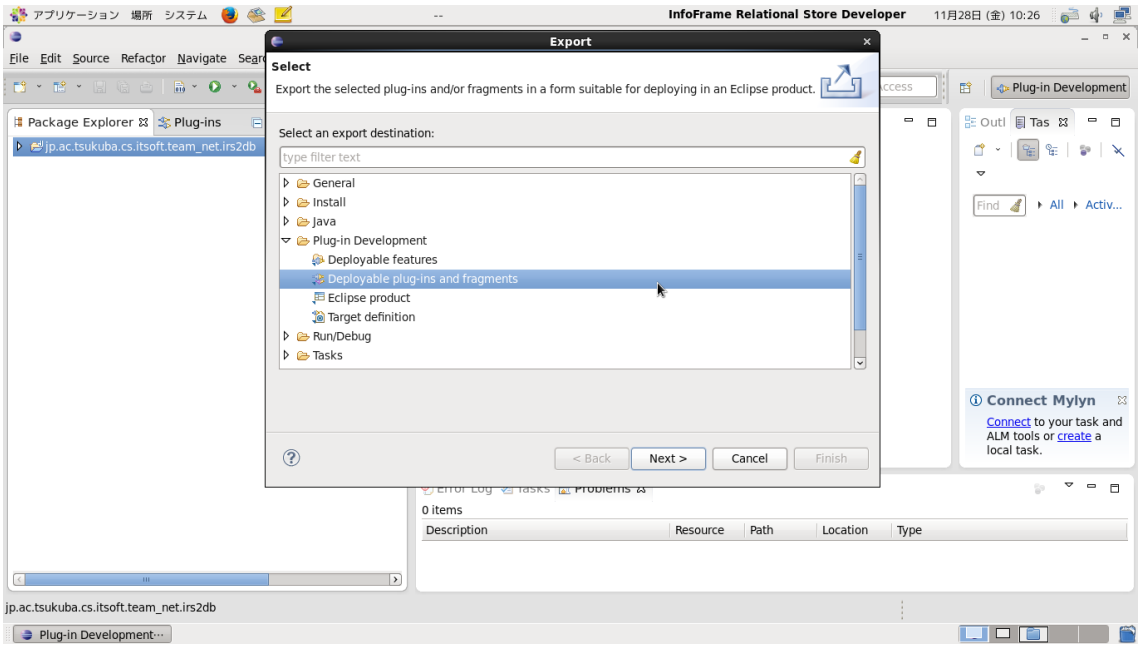
5. NET ツールのプロジェクトをインポート成功するとこのような画面になる。



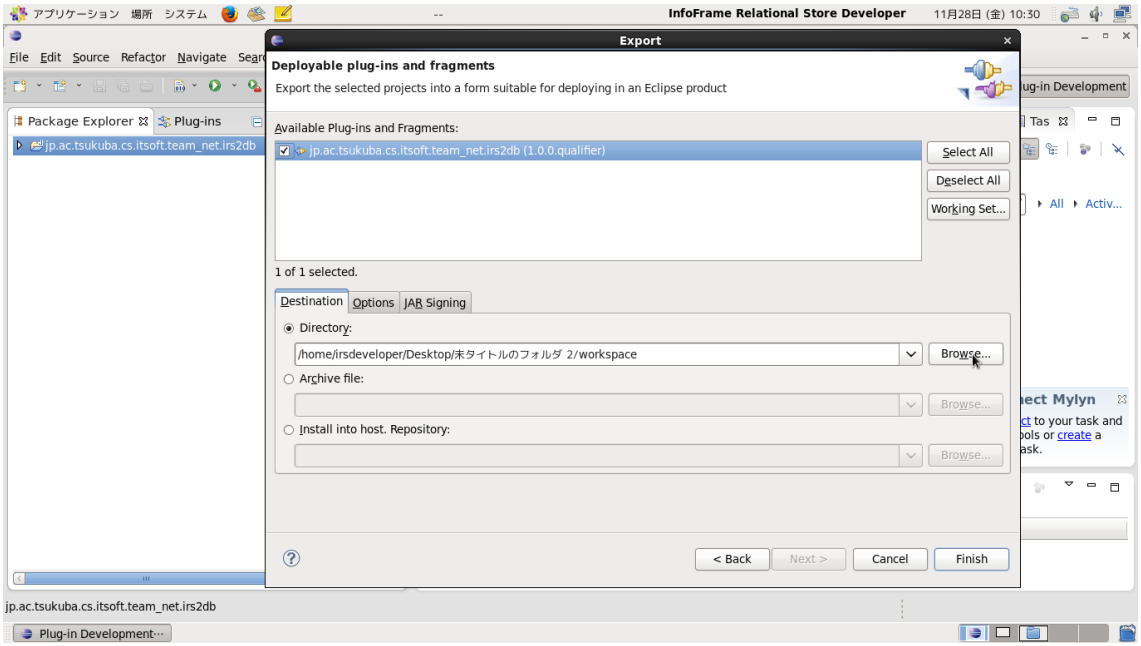
6. NET ツールのプロジェクト右クリックし、「Export」を選択する。



7. 「Deployable plug-ins and fragments」を選択する。

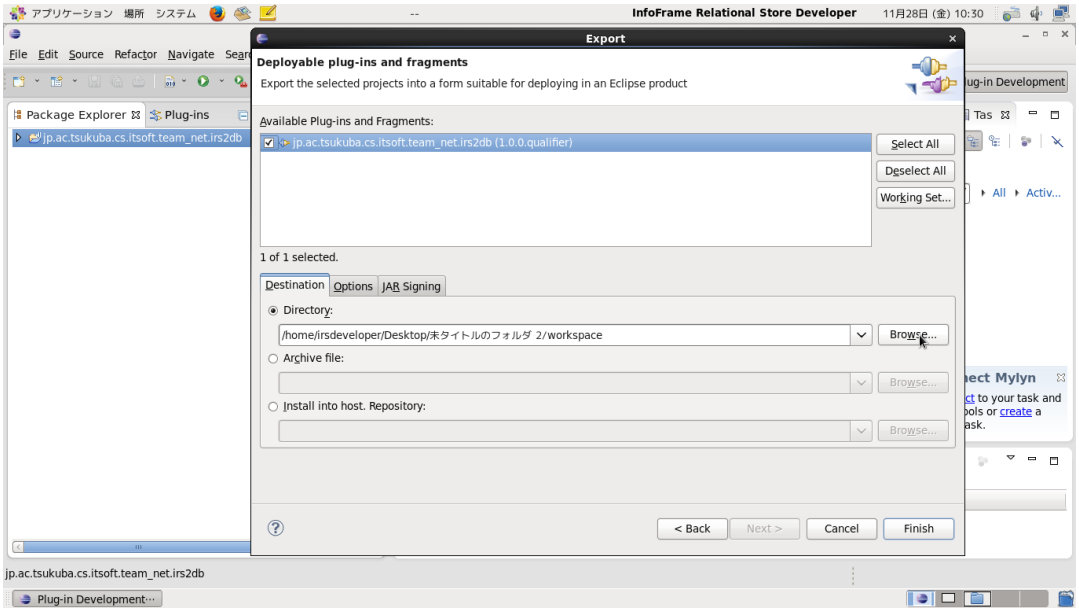


8. NET ツールのプロジェクトを選択する。



以下、３枚の画像は「Deployable plug-ins and fragments」の設定画面である。

8.1「Destination」の Directory には NET ツールプラグインの jar ファイルが出力する場所を入力する。

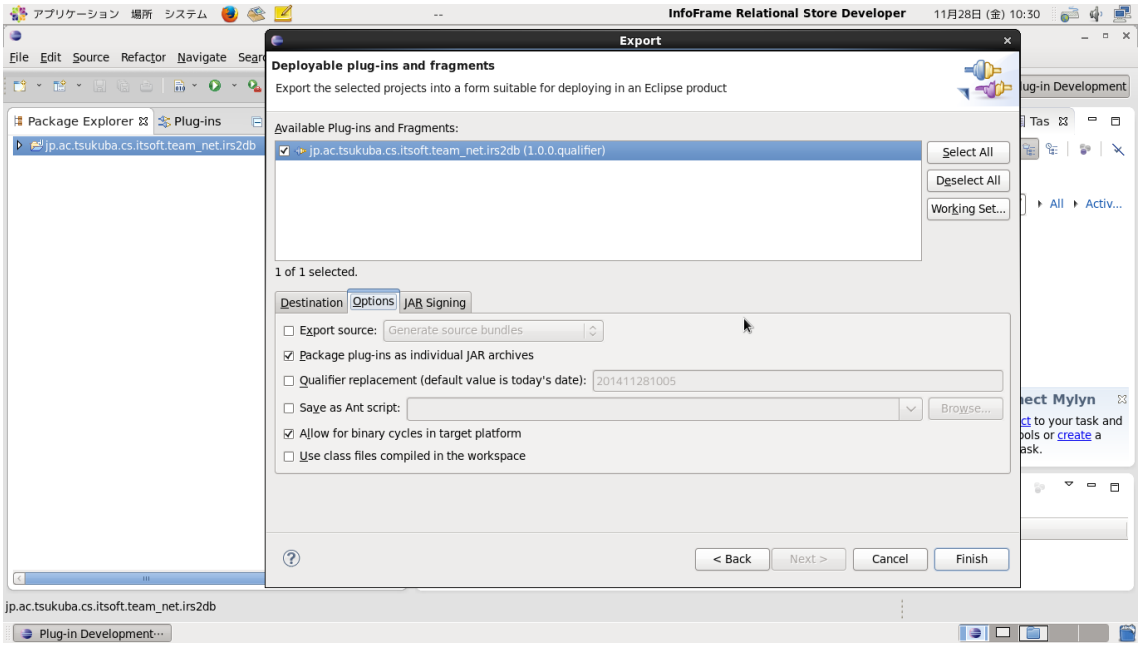


8.2「Options」は以下のように設定する。

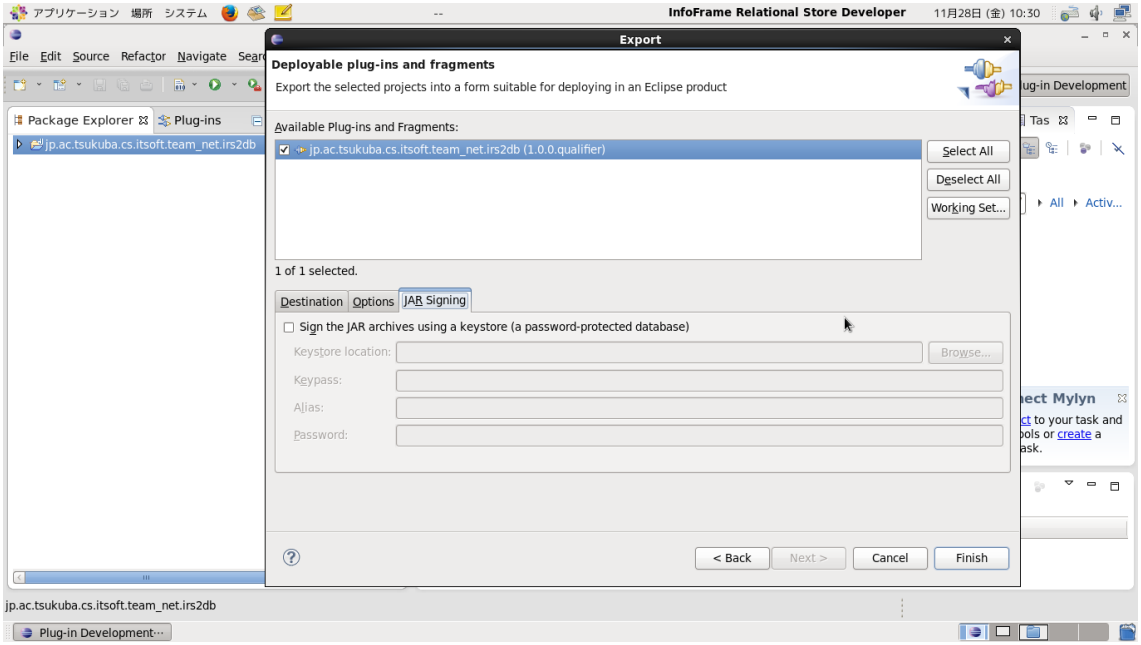
Package plug-ins as individual JAR archives

Allow for binary cycles in target platform

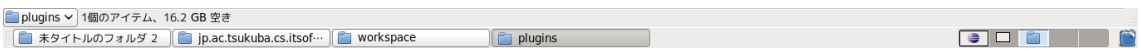
のチェックを入れる。



8.3「JAR Signing」は以下のように設定する。(変更項目はないと思われる)



8 の設定終了後に右下の「Finish」ボタンを押します。NET ツールのプラグインのビルドが始まる。完了すると指定したフォルダに NET ツールのプラグインが生成されている。



生成したプラグインファイルの使用方法は８章３節「セットアップマニュアル」を参照。

第４節　　ツールを導入したEclipse

本ツールのプラグインを導入済みのEclipseを納品フォルダ内の
eclipse_plugin_includeフォルダ内に配置した。

Linux 64-bit用

Eclipse for RCP and RAP Developers

Version: Kepler Service Release 2

Build id: 20140224-0627

http://www.eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/keplersr2　からダウンロードしたものを使用した。

	質問事項	回答	自由記述欄(任意)
0(説明)	質問事項です、質問を読んで右の回答欄に回答をお願いします。文頭の(選択)は回答が選択式であることを示しています。(自由記述)は回答が自由記述となっております。	回答欄です。選択式の場合は回答を選択してください。自由記述式の場合は入力をお願いします。文字数制限はありません。	自由記述欄です。質問事項や回答についてなにかあれば入力をお願いします。任意となっておりますので、ない場合は空欄で構いません。
1	(選択)実際出来上がったGUIと設計はどの程度一致していましたか	(回答を選択してください)	
2	(選択)NETツールのデザインとEclipseデザインで違和感を感じることはありましたか	(回答を選択してください)	
3	(選択)NETツールの操作はEclipseの操作と大きな違いがありましたか	(回答を選択してください)	
4	(選択)NETツールの操作で違和感を覚えることはありませんでしたか	(回答を選択してください)	
5	(選択)4で違和感を覚えると答えた場合、違和感を覚えた操作はどれですか	(回答を選択してください)	
6	(選択)NETツールの操作は簡単でしたか	(回答を選択してください)	
7	(選択)NETツールのGUIの満足度はどの程度ですか	(回答を選択してください)	
8	(自由記述)GUIで改善してほしい点がありますか	(自由記述回答)	

	質問事項	回答	自由記述欄(任意)
0(説明)	質問事項です、質問を読んで右の回答欄に回答をお願いします。文頭の(選択)は回答が選択式であることを示しています。(自由記述)は回答が自由記述となっております	回答欄です。選択式の場合は回答を選択してください。自由記述式の場合は入力をお願いします。文字数制限はありません。	自由記述欄です。質問事項や回答についてなにかあれば入力をお願いします。任意となっておりますので、ない場合は空欄で構いません。
1	(選択)最初に期待していた結果と最終的に実現した内容の一致度はどれくらいですか。理由も自由記述欄にお願いします	(回答を選択してください)	
2	(選択)本プロジェクトの総合満足度はいかがでしたか。理由も自由記述欄にお願いします	(回答を選択してください)	
3	(自由記述)プロジェクト運営として一番課題であると感じた点は何ですか		
4	(自由記述)プロジェクト運営として評価できることはありましたか、あった場合はそれは何ですか		
5	(選択)NETツールを用いることで、どの程度データ作業時のユーザーの負担は軽減されると思いますか	(回答を選択してください)	
6	(自由記述)進捗会議で、「プロジェクトの進捗を知ることで、安心感がある。」など利点はありましたか。利点を回答欄に記述してください		
7	(自由記述)今後も開発が続くとした場合、進捗会議の頻度(1週間に1度)は適切だと思いますか。理由も添えてお願いします		
8	(選択)進捗会議でRedmineのチケットを用いて説明しましたが、進捗会議の資料として適当だと思いましたか	(回答を選択してください)	
9	(自由記述)今回のプロジェクト運営についてコメントが有りましたらお願いします		

質問事項					
データベースの追加	100%	大きな違いがあった	かなり違和感があった	難しい	思わない
データベースへの接続	90%	少し違いがあった	少し違和感があった	やや難しい	やや思わない
テーブル情報の確認	80%	あまり違いはなかった	あまり違和感はなかった	どちらでもない	どちらでもない
テーブルの作成	70%	まったく違いはなかった	まったく違和感はなかった	やや簡単	やや思う
テーブル、関数の対応付け	60%	(回答を選択してください)	(回答を選択してください)	簡単	思う
抽出条件の設定	50%			(回答を選択してください)	(回答を選択してください)
変換、マッピング条件の設定	40%				
ETL実行	30%				
その他	20%				
(回答を選択してください)	10%				
	0%				
	(回答を選択してください)				