Alibaba Cloud DataWorks

データの開発

Document Version20191205

目次

1 データの開発1
1.1 ソリューション1
1.2 SOL コードに関するエンコードの原則と基準3
1.3 コンソールの機能
1.3.1 コンソールの紹介9
1.3.2 バージョン11
1.3.3 構造12
1.3.4 リレーションシップ15
1.4 ノードタイプ16
1.4.1 ノードタイプの概要16
1.4.2 データ統合ノード17
1.4.3 ODPS SQL ノード18
1.4.4 SQL コンポーネントノード
1.4.5 仮想ノード
1.4.6 ODPS MR ノード 29
1.4.7 SHELL ノード35
1.4.8 PyODPS ノード
1.4.9 クロステナントノード 42
1.4.10 マージノード (Merge node)45
1.4.11 分岐ノード (Branch node) 50
1.4.12 割り当てノード (Assignment node)56
1.5 スケジューリングの設定 59
1.5.1 基本属性59
1.5.2 パラメーターの設定61
1.5.3 時間属性68
1.5.4 ノードコンテキスト 77
1.6 設定管理
1.6.1 設定管理の概要82
1.6.2 設定センター
1.6.3 プロジェクト設定
1.6.4 テンプレート
1.6.5 テーマの管理
1.6.6 テーブルレベル
1.7 マニュアルビジネスフロー89
1.7.1 マニュアルビジネスフローの紹介89
1.7.2 リソース
1.7.3 関数94
1.7.4 テーブル
1.8 マニュアルタスクのノードタイプ103
1.8.1 仮想ノード
1.8.2 SQL コンボーネントノード105

	1.8.3 ODPS MR ノード	110
	1.8.4 マニュアルデータ統合ノード	116
	1.8.5 PyODPS ノード	
	1.8.6 ODPS SQL ノード	125
	1.8.7 SHELL ノード	
	1.9 マニュアルタスクパラメーターの設定	130
	1.9.1 基本属性	
	1.9.2 マニュアルノードパラメーターの設定	
	1.10 コンポーネントの管理	
	1.10.1 コンポーネントの使用	138
	1.10.2 コンポーネントの作成	139
	1.11 クエリ	146
	1.12 ランニングログ	149
	1.13 パブリックテーブル (Public Table)	150
	1.14 テーブルの管理	152
	1.15 外部テーブル	158
	1.16 関数	170
	1.17 エディターショートカット一覧	171
	1.18 ゴミ箱	174
2 D	ataService Studio	176
2 D	2 1 DataSamiaa Studio の概要	176
	2.1 DataService Studio の	170
	2.2 用田来 2.2 ADI の生成	1//
	2.5 API の主版	1/0
	2.3.1 アーダノーへの設定 2.2.2 ADI 生式の概要	170
	2.3.2 API 主风の概安	178
	2.3.3 ワイサートモート CO API の生成	179
	2.3.4 スクリノトモート Cの API の主成	184
	2.4 API の公開	190
	2.5 API の削除	
	2.6 API の呼び出し	
	2./ よくある質問	193

1**データの開発**

1.1 ソリューション

データ開発モードは、3 階層構造 (プロジェクト - ソリューション - ビジネスフロー) ヘアップグ レードされました。従来のディレクトリ編成モードは使用しません。

プロジェクト - ソリューション - ビジネスフロー (Project - solution - business flow)

DataWorks の新バージョンでは、ビジネスタイプに基づいて、異なるタイプのノードタスクを 統合できるように、データ開発モードがアップグレードされました。 このような構造を使用する ことで、ビジネスごとのコード開発がさらに容易になります。 開発プロセスでは、より広い視 野で複数のビジネスフローにまたがる開発が実現します。 ユーザーの開発エクスペリエンスを 向上させるために、開発プロセスはプロジェクト - ソリューション - ビジネスフロー (Project solution - business flow) の 3 階層構造に基づいて再定義されます。

- ・ プロジェクト:権限組織の基本単位で、ユーザーの権限 (開発や O&M 権限) を制御するために
 使用します。同じプロジェクト内では、プロジェクトメンバーのすべてのコードを共同で開発、管理することができます。
- ・ ソリューション: ビジネスフローを組み合わせて、ソリューションをカスタマイズします。利
 点:
 - 1つのソリューションが複数のビジネスフローから構成されます。
 - 同じビジネスフローを別のソリューションで再利用することができます。
 - 没入型開発を組み合わせたソリューションへ実装することができます。
- ・ビジネスフロー:ビジネスの抽象エンティティです。ビジネスの視点でデータコード開発を整 理することができます。ビジネスフローは、複数のソリューションで再利用できます。利点:
 - ビジネスフローでは、ビジネスの視点からコードを整理することができます。 タスクタイ プベースのコード編成モードがあります。 複数階層のサブディレクトリをサポートします(4 階層までを推奨)。
 - ビジネスの視点からワークフロー全体を確認し、最適化することができます。
 - ビジネスフローダッシュボードを使用することで、開発効率が向上します。
 - ビジネスフローに基づいて、リリースや O&M を編成することができます。

没入型開発エクスペリエンス

作成したソリューションをダブルクリックすると、開発エリアからソリューションエリアへ切り 替わります。 現在のソリューションの内容のみが、ディレクトリに表示されます。 最新の環境が 提供されるため、現在のソリューションと関連のないプロジェクトのコードに煩わされることが ありません。

1. [DataStudio] ページへ移動し、ソリューションを作成します。



2. 作成したソリューションから閲覧するビジネスフローを選択します。



3. 選択したビジネスフローのノードを閲覧する場合、またはソリューションを変更する場合 は、[View All Business Flows] を右クリックします。

Ш	Data Developn 온 🗟 다 C 🕀	Ъ	Business Flow 🗙	Solution	× Sq rpt	_user_info_d ×	Sq dw_user_info_all_d x	Sq ods_log_info_d ×	Fx getregion	×
873		T								
*	✓ Solution									
民	What is the solution? Create one.		Create Business Fl	low		base_cdp		workshop		
ė.	> Business Flow	₽₽								
۵			View All Business Flow	rs						
#			View All Busin	ness Flows						
R										
52										
Ť			works							
			-							
۵										

- 4. 別のページへ移動します。
 - [Publish] をクリックすると、[Task Publish]ページへ移動します。現在のソリューションで [To be released] となっているノードがこのページに表示されます。
 - [O&M] をクリックすると、[O&M Center] > [Periodic Instances] ページへ移動します。現在のソリューションにあるすべてのノードの周期インスタンスがデフォルトでこのページに表示されます。

ビジネスフローは、複数のソリューションで再利用できます。 独自のソリューションの開発に 集中することができます。 他のソリューションやビジネスフロー内で参照したソリューション の編集を別のユーザーが直接行うことができ、共同開発が実現します。

1.2 SQL コードに関するエンコードの原則と基準

エンコードの原則

SQL コードは次のとおりエンコードします。

- ・ コーディングの原則
- コード行は明確で、きちんとした見栄えのものにします。
- コード行はきれいに配置し、正しい階層構造にします。
- コードの可読性を向上させるために、必要に応じてコメントを入力します。

- この規則の要件は開発者のコーディング作業を制約するものではありません。実際には、一般的な要件に違反がないという前提で、コード開発にとってメリットがある場合は、この規則から合理的に逸脱しても構いません。この規則は、継続して改善および補足される予定です。
- SQL コードに使用するキーワードや予約済みワードには小文字を使用します。これらのワードの例としては、select、from、where、and、or、union、insert、delete、group、having および count があります。
- ・ SQL コードに使用するキーワードと予約済みコードに加えて、その他のコード (フィールド名 やテーブル別名など) も小文字にします。
- ・スペース4つ分は、1インデント単位と同等です。すべてのインデントはインデント単位の倍数の整数とし、コードの階層に応じて整列させます。
- "select * operation" の使用は禁止されています。列名はすべての操作で指定する必要があります。
- ・ 対応するラケットは同じ列に記述する必要があります。

SQL コーディングの仕様

SQL コードの仕様は次のとおりです。

・ コードヘッダー

情報カテゴリ、関数の説明、作成者、日付などの情報をコードヘッダーへ追加します。 ログと タイトルバーは、後にユーザーがレコードを変更する際に追加できるよう確保しておきます。 各行は、80 文字を超えることができません。 テンプレートは次のとおりです。

 ****	***	***	***	****	*****
 **	Subject:	AGDS Risk ap	plication		
 **	function description:	Credit index in	terface		
 **	creator:	chenfeng			
 **	create date:	2014-05-23			
 **	Modify the log:				
 **	Modify the date	21	Modifier		Modifies the content
 **	2014-05-23	}	chenfeng		create
 ****	*****	*****	******	*****	*****

- ・ フィールドの配置要件
 - "select" 文で選択するフィールドに対して、フィールドごとに1行使用します。
 - **"select"**の次にインデントを1つ入れてから、最初に選択したフィールドを続けます。 したがって、行の先頭から インデント2つ分先からフィールドを記述します。
 - その他の各フィールドは、インデント2つ目から始め、コンマ(,)の後にフィールド名を記述します。
 - 2つのフィールド名の間にあるコンマ(,)は、2つ目のフィールドの直前に記述します。
 - "as" 文は、関連するフィールドと同じ行に記述します。 複数のフィールドに "as" 文を使用 する場合、同じ列に配置することを推奨します。

select	<pre>channel_id , trade_channel_desc , trade_channel_edesc , inst_date , trade_iswap , channel_type , channel_second_desc</pre>	as channel_id as trade_channel_desc as trade_channel_edesc as inst_date as trade_iswap as channel_type as channel_second_desc
from	(

・ "Insert" サブ文の配置要件

"Insert" サブ文は同じ行に記述する必要があります。 改行は禁止されています。

・ サブ文の配置要件を選択します。

"select" 文に使用されるサブ文 (from、where、group by、having、order by、join、 および union) は、次の要件に準拠する必要があります。

- 改行します。
- サブ文は、"select" 文に対して左揃えにします。
- サブ文とその後に続くコードの最初の文字の間にインデントを2つ入れます。
- "where" サブ文の論理演算子 ("and" や "or") は where に対して左揃えにします。
- サブ文の長さがインデント2つ分を超える場合、サブ文の後にスペースを追加し、後続の コード ("order by" や "group by") を記述します。

select	trim(channel) channel min(id) id
from where and and group by order by	<pre>ods_trd_trade_base_dd channel is not null dt = \${tmp_uuuummdd} trim(channel) <> '' trim(channel)</pre>

・ 演算子の前後につけるスペースの要件は、算術演算子と論理演算子の前後に1つずつ入れる必要があります。1行の長さが80文字を超えない限り、演算子は同じ行に記述します。

select	trim(channel) channel
	,min(id) id
from	ods_trd_trade_base_dd
where	channel is not null
and	dt = \${tmp_uuuummdd}
and	trim(channel)_<>_''
group by	trim(channel)
order by	trim(channel)

・ "case" 文の記述

"select" 文では "case" 文を使用してフィールド値を判断したり割り当てます。 コード行の可 読性を向上させるためには、"case" 文を正しく記述することが重要です。

以下は、"case" 文の記述に関する規則です。

- "when" サブ文は、 "case" 文と同じ行に記述し、インデントを1つ入れます。
- **"when"** サブ文ごとに1行使用します。 文が長すぎる場合は、改行を行うことができます。
- "case" 文には、"else" サブ文が含める必要があります。 "else" サブ文は、"when" サブ 文に対して左揃えにします。

, case when pl.trade_from = '3008' and pl.trade_email is null then 2
when pl.trade_from = '4000' and pl.trade_email is null then 1
when p9.trade_from_id is not null then p9.trade_from_id
end
,pl.trade_email as partner_id

入れ子クエリの記述に関する仕様

入れ子サブクエリは、データウェアハウスシステムの ETL 環境でよく使用されます。した がって、階層状にコードを配置することが重要です。例:

select	p. channel , rownumber()	order i	d		
from	(
	select	sl. chan	inel		
	101101-001	,s1.id			
	from	(15		44 63
	-		select	trim(channel) ,min(id)	as channel as id
			from	ods_trd_trade_base	e_dd
			where	channel is not nul	11
			and	dt = \${tmp_yyymm	dd}
			and	trim(channel) \Diamond	
) [1	group b	y trim(channel)	
	left out	er join			
	icit out	dim trad	le chann	el s?	
	on	s1 chann	el = s2	trade channel ede	sc
	where	s2. trade	channe	l edesc is null	
	order by	id	-		
) p				
;					

- ・ テーブル別名の定義規則
 - 別名をすべてのテーブルに追加する必要があります。"select" 文で操作テーブルに対し別
 名を定義した後、そのテーブルを参照する文を記述する際、常にその別名を使用する必要
 があります。コードの記述を容易に行うには、別名は可能なかぎりシンプルで簡潔なもの
 にし、キーワードの使用は避けます。
 - テーブルの別名はシンプルな文字を使って定義します。別名はアルファベット順に定義することを推奨します。
 - 別名の多層入れ子サブクエリの前に、階層を示す必要があります。 SQL 文の別名はレイ ヤー別に定義します。レイヤー1から4まではそれぞれP(Part)、S(Segment)、U (Unit)、D(Detail)と表現します。あるいは、レイヤー1から4までを、a、b、c、dと 表現することもできます。同じレイヤーにあるサブ文を区別するために、レイヤーを示す

文字の後ろに番号 (1、2、3、4 など) を付けます。 必要に応じて、テーブルの別名にコメ ントを追加することができます。

```
p. channel
select
             , rownumber()
                               order_id
from
                    select
                               . cna
, sl. id
(
                               s1. channel
                    from
                                        select trim(channel)
                                                                          as channel
                                                  , min(id)
                                                                          as id
                                                  ods_trd_trade_base_dd
                                        from
                                                  channel is not null
dt = ${tmp_yyyymmdd}
                                        where
                                        and
                                                  trim(channel) ↔
                                        and
                                        group by trim(channel)
                   ) sl
left outer join
                              dim_trade_channel s2
                             s1.channel = s2.trade_channel_edesc
s2.trade_channel_edesc is null
                    on
                    where
                    order by id
             ) p
-
```

- ・ SQL コメント
 - 各 SQL 文に対してコメントを追加する必要があります。
 - 各 SQL 文のコメント用に1行使用し、文の前に配置します。
 - フィールドのコメントは、フィールドの次に記述します。
 - 簡単に理解できない分岐条件の場合は、コメントを追加します。
 - 関数を記述するための重要な計算に対してもコメントを追加します。
 - 関数が長すぎる場合は、実装する関数に基づいて文をセグメント化し、それぞれのセグメ ントを説明するコメントを追加します。
 - 定数または変数の場合は、保存する値の意味を説明するコメントを追加します。値の有効
 範囲を説明するコメントも任意に追加できます。

1.3 コンソールの機能

1.3.1 コンソールの紹介

		Data Development	ARDCOA	ereste_dd ×	
	Data Development			3 분 배 때 ☆ 이 편 및 이 은 전 및 문 분 비 22 Publish 23	
*	Components	> Solution	88	1 招加业务职业中国省的表 (1877-	Se
R	Queries	✓ Business Flow	88	CREATE TABLE IF NOT EXISTS movies_1	hedule
B	Runtime Log	> 基 推荐引擎workshop > 基 DataWorks_Test		5 movieid BIGINT 785.	e R
	Manual Business Flow	∽ 🚣 Movies_ODS		7 ,genres STRING	elation
E	Public Tables	> 😑 Data Integration > 🕂 Data Developme	nt		nship
R	Tables	> 🔲 Table		11 CREATE TABLE IF NOT EXISTS tags_1 12 (Ver
5	Functions	> 🔗 Resource		13 userid BIGINT 14 ,movieid BIGINT	
t	Recycle Bin	> 🔚 Algorithm		tag STRING to ,tp STRING	
		> 💽 control > 🚑 shenyun828		17) 18 ; 19	cture

インターフェイス機能についての説明は次のとおりです。

No.	機能	説明
1	Show My Files	クリックして、現在の列で、ご自身のアカウントにある ノードを表示します。
2	Code Search	クリックして、コードまたはコードセグメントを検索し ます。
3	[+]	クリックして、ソリューション、ビジネスフロー、フォ ルダ、ノード、テーブル、リソース、または関数エント リを作成します。
4	Reload	クリックして、現在のディレクトリツリーを更新しま す。
5	Locate	クリックして、選択したファイルの場所を特定します。
6	Import	クリックして、ローカルデータをオンラインテーブルへ インポートします。 エンコード形式に注意します。
7	Filter	クリックして、指定した条件に基づいてノードをフィル タリングします。
8	Save	クリックして、現在のコードを保存します。
9	Save as Query File	クリックして、現在のコードを一時ファイルとして保存 します。このファイルは [Temporary query] 列に表示 されます。
10	Submit	クリックして、現在のノードを送信します。

No.	機能	説明
11	Submit and Unlock	クリックして、現在のノードを送信し、コードを編集で きるようにロックを解除します。
12	Steallock	このコードのオーナーではない場合に、クリックして ノードを編集します。
13	Run	クリックして、現在のノードのコードを実行します。
14	Run After Setting Parameters	クリックして、設定したパラメーターを使って現在の ノードのコードを実行します。
15	Precompile	クリックして、現在のノードのパラメーターを編集およ びテストします。
16	Stop Run	クリックして、現在実行中にコードを停止します。
17	Reload	クリックして、ページを更新し、以前保存したページへ 戻ります。
18	Run Smoke Test in Development Environment	クリックして、開発環境にある現在のノードのコードを テストします。
19	View Smoke Test Log in Development Environment	クリックして、開発環境で実行されているノードの実行 ログを表示します。
20	Go to Scheduling System of Development Environment	クリックして、開発環境の O&M センターへ移動しま す。
21	Format	クリックして、現在のノードのコードを配列します。 コードを1 行におさめるには長すぎる場合に使用しま す。
22	Publish	クリックして、送信したコードを公開します。 コードが 公開された後、コードは運用環境にあります。
23	O&M	クリックして、運用環境の O&M センターへ移動しま す。
24	Scheduling Configuration	クリックして、ノードのスケジューリング属性、パラ メーター、およびリソースグループを設定します。
25	Relationship	クリックして、コードで使用されるテーブルの関係を表 示します。
26	Version	クリックして、現在のノードの送信レコードと公開レ コードを表示します。

No.	機能	説明
27	Structure	クリックして、現在のノードのコード構造を表示しま す。 コードが長すぎる場合、構造のキー情報を元にコー ドセグメントを簡単に検索することができます。

1.3.2 バージョン

バージョンとは、現在のノード送信レコードとリリースレコードです。送信を行うたびに新しい バージョンが生成されます。 ノードの操作を簡単にするために、必要に応じて関連ステータスの 確認、タイプの変更、備考のリリースを行うことができます。

注
任

•

送信されたノードのみにバージョン情報が存在します。

5000118 87	V7	dataworks_dem o2	2018-09-02 10:3 9:57	Edit	Publish ed	test	View Code Roll Back	hedule
5000118 87	V6	dataworks_dem o2	2018-09-02 10:3 7:47	Edit	Publish ed	123	View Code Roll Back	Relationship
5000118 87	V5	dataworks_dem o2	2018-09-02 10:3 6:28	Edit	Publish ed	test		
5000118 87	٧4	dataworks_dem o2	2018-09-02 10:3 3:54	Edit	Publish ed	test	View Code Roll Back	Structure
5000118 87	V3	dataworks_dem o2	2018-09-02 10:3 0:19	Edit	Publish ed	test	View Code Roll Back	
5000118 87	V2	wangdan	2018-08-31 10:2 1:19	Edit	Publish ed	workshop user portrait part is w ritten logically.	View Code Roll Beck	
5000118 87	VI	wangdan	2018-08-30 17:3 7:55	Add	Publish ed	workshop user portrait part is w ritten logically.	View Code Roll Back	

- ・ File ID: 現在のノード ID
- Version: リリースするたびに新しいバージョンが生成されます。1回目のリリースはV1、2回目の編集ではV2などです。
- · Submitter: ノードの送信やリリース操作を行ったユーザー
- Submission Time: バージョンのリリース時間 バージョンが送信後にリリースされた場合、 リリース時間が送信時間となります。デフォルトでは、操作の最終リリース時間が記録されま す。
- Change Type: 現在のノードの操作履歴 ノードの1 回目のリリースの場合、[Added] と設定 されます。ノードが編集された場合は、[Modified] と設定されます。

- · Status: 現在のノードの操作ステータスレコードです。
- ・ Remarks: 送信される際の現在のノードの説明を変更します。 ノードを操作中に他の担当者 が関連したバージョンを探す際に役立ちます。
- · Action: この列では、[Code] と [Roll Back] を選択できます。
 - View Code: クリックしてバージョンコードを表示し、元に戻すレコードバージョンを正確に検索します。
 - Roll Back: クリックして現在のノードを必要に応じて以前のバージョンへ元に戻します。 元に戻した後、ノードをリリースするために再度送信する必要があります。
- · Compare: クリックして2つのバージョンのコードとパラメーターを比較します。



[View Details] をクリックし、詳細ページへ移動します。コードとスケジューリング属性の 変更を比較します。

门注:

2つのバージョンでのみ比較することができます。3つ以上のノードは比較できません。

1.3.3 構造

構造は現行コードに基づいており、SQL で実行されるプロセス図が記述されます。編集した SQL を即座に確認できるため、編集および表示を簡単に行えます。

構造

SQL は次のとおりです。

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.
bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
, b.gender
, b.age_range
```

, B. flavdiac , a.region , a.device , a.identity , a.method , a.url , a.referer , a.time FROM (VALUES From fig WHERE dt = \${bdp.system.bizdate}) a LEFT OUTER JOIN (VALUES FROM ods_user_info_d WHERE dt = \${bdp.system.bizdate}) b on a.uid = b.uid ;

このコードによると、構造は次のように説明されます。



円にマウスを置くと、対応する説明が表示されます。

- 1. Source table: SELECT クエリのターゲットテーブルです。
- 2. Filter: 照会するテーブルの特定のパーティションにフィルターをかけます。
- 3. 中間のテーブル (クエリビュー) の1 番目の部分: 一時テーブルヘクエリデータの結果を配置します。
- 4. Join: 結合機能を使って 2 部クエリの結果をモザイクします。
- 5.2番目のセクションにある中間テーブル (クエリビュー): 結合結果を一時テーブルへ要約しま す。この一時テーブルは、3日保管され、3日後に自動的にクリアされます。
- 6. Target table (insert): 2 番目で取得されたデータをテーブルへ挿入し上書きします。

1.3.4 リレーションシップ

類似リレーションは、現在のノードとその他のノードのリレーションシップを示します。 リレー ションシップには2つの部分、依存ダイアグラムと内部リレーションシップマップがあります。

依存グラフ

ノードの依存性によって、依存グラフは現在のノードが想定しているものかどうかを示します。 想定しているものではない場合、Schedule 設定インターフェイスへ戻りリセットします。



内部リレーションシップマップ

内部リレーションシップマップは、ノードのコードに基づいて説明します。たとえば

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.
bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , B. flavdiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
 VALUES
 From fig
 WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
 VALUES
  FROM ods_user_info_d
 WHERE dt = ${bdp.system.bizdate}
) b
```

```
on a.uid = b.uid ;
```

SQL によると、次の内部リレーションシップマップは解決され、テーブル間のリレーションシップを示すために結合モザイクとして使用されるアウトプットテーブルを説明します。



1.4 ノードタイプ

1.4.1 ノードタイプの概要

DataWorks では、7 つのタイプのノードが提供されます。異なる使用ケースが適用されます。

仮想ノードタスク

仮想ノードは、データを生成しない制御ノードです。 通常、ワークフローでのノードの全体的 な計画に対して、ルートノードを使用します。 仮想ノードタスクについての詳細は、「仮想ノー ド」をご参照ください。

注注:

ワークフローの最終出力テーブルには、複数の分岐入力テーブルが含まれます。 仮想ノードは 通常、これらテーブルに依存関係がない場合に使用されます。

ODPS SQL **タスク**

ODPS SQL タスクでは、Web 上で SQL コードを直接編集したり管理することができます。また、簡単に実装の実行やデバッグ、共同開発を行うことができます。 DataWorkd では、コード バージョンの管理、上位および下位の依存関係の自動解決などの機能が提供されます。 例の詳細 については、「ODPS SQL ノード」をご参照ください。 DataWorks では、デフォルトで MaxCompute のプロジェクトが開発および運用スペースとし て使用されます。このため、ODPS SQL ノードのコードは、MaxCompute SQL 構文に従いま す。MaxCompute SQL 構文は、Hive 構文のように標準の SQL のサブセットとして使用でき ます。しかし、MaxCompute SQL はデータベースと同一視することはできません。これは、 トランザクション、主キーの制約、インデックスといったデータベースにある機能の多くが、 MaxCompute SQL にはないためです。

特定のMaxCompute SQL 構文についての詳細は、「SQLの概要」をご参照ください。

ODPS MR **タスク**

MaxCompute では、MapReduce プログラミング API がサポートされています。これらの Java API は、MaxCompute のデータプロセスで MapReduce プログラムをコンパイルするた めに使用できます。 ODPS MR ノードを作成しタスクスケジューリングに使用することができま す。例の詳細については、「*ODPS MR ノー*ド」をご参照ください。

PyODPS **タスク**

MaxCompute では Python SDK が提供されます。これを使って MaxCompute を操作します。

DataWorks では、PyODPS タスクタイプが提供され、 MaxCompute の Python SDK を 統合することができます。 Python コードを直接編集し、DataWorks の PyODPS ノードの MaxCompute を操作することができます。 詳細は、「*PyODPS ノ*ード」をご参照ください。

SQL コンポーネントノード

SQL コンポーネントノードは、SQL コードプロセステンプレートで、複数の入出力パラメー ターが含まれます。SQL コードプロセスを操作するには、1 つ以上のソースデータテーブルをイ ンポートしフィルターをかけます。結合して集約し、新しいビジネスに必要なターゲットテーブ ルを形成します。詳細は、「SQL コンポーネントノード」をご参照ください。

データ同期タスク

データ同期ノードタスクは、Alibaba Cloud DTplus プラットフォームで提供される安定性と 効率性に優れ、また自動的でスケーラブルな外部データ同期クラウドサービスです。 データ同期 ノードを使うと、ビジネスシステムのデータを MaxCompute へ簡単に同期することができま す。詳細は、「データ統合ノード」をご参照ください。

1.4.2 データ統合ノード

現在、データ統合タスクでサポートされているデータソースは、MySQL、 DRDS、SQL Server、PostgreSQL、 Oracle、 MongoDB、 DB2、 OTS、 OTS Stream、 OSS、 FTP、 Hbase、LogHub、HDFS、および Stream です。 サポートされているデータソースについての詳細は、「#unique_17」をご参照ください。

統合タスクの設定

詳細は、次をご参照ください。 #unique_18/unique_18_Connect_42_section_tfn_1kc_p2b

ノードスケジューリングの設定

ノードタスクの編集エリアの右側にある [Scheduling Configuration] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリング設定」をご参照ください。

ノードの送信

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を押し てノードを開発環境へ送信 (ロックを解除) します。

ノードタスクの公開

操作の詳細については、「リリース管理」をご参照ください。

本番環境でのテスト

詳細は、「#unique_20」をご参照ください。

1.4.3 ODPS SQL **ノード**

ODPS SQL 構文は、SQL 構文と同じです。データは大容量 (TB 級) だが、リアルタイム要件 が高くない分散シナリオに適用されます。 スループット向けの OLAP アプリケーションです。 ジョブの準備から送信までのプロセスが完了するのに長時間かかるため、ビジネスで数万件のト ランザクション処理が必要な場合に ODPS SQL の使用を推奨します。 1. ビジネスフローを作成します。

[Data Development] の [Business Flow] を右クリックし、[Create Business Flow] を選 択します。

	Data Developn 🖉 🛱	📮 C' 🕀 🗗 🧰 Data Developr
(/)	Enter a file or creator name	Solution New
*	> Solution	Business Flow New
M	➤ Business Flow	Folder Business Flow
	> 嚞 数据汇总	Data Integration >
	> 矗 数据清洗_1	Data Development 🔸
	> 晶 数据清洗_2	Table
Ħ	> 晶 数据清洗_3	Resource >
	> 🎝 数据清洗_4	
B	> 📇 Bl_Demo	Function
	🗸 🛃 BirdLiu	
	> Data Integrati	ion

2. ODPS SQL ノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [ODPS SQL] の順に選択します。



3. ノードコードを編集します。

SQL 文の構文についての詳細は、「*MaxCompute SQL* 文」をご参照ください。

Data Developn 온 菣 다 C 🕀 🕁	Sq insert_data ×
Enter a file or creator name	🖱 🖽 G 🔒 💿 :
> Solution	
➤ Business Flow	
✔ 🛃 base_cdp	4create time:2018-08-31 15:59:06
🔉 ≓ Data Integration	
👻 🚺 Data Development	
• 🧐 insert_data 🗄 🖬 🖅 08-31	
● 🔽 start 王丹锁定 08-31 15:58	

4. クエリ結果の表示

DataWorks のクエリ結果は、スプレッドシート機能に連携されるため、データ結果を簡単に 操作できます。

クエリ結果は、直接スプレッドシート形式で表示されます。 DataWorks で操作を行ったり、 スプレッドシートを開いたり、ローカルのエクセルへコンテンツステーションを自由にコピー できます。

Sq O	ds_log_info_d	× Sq	dw_user	_info_all_d	×	Sq cr	eate_table_	,ddl (Di write_	esult
۳	E 6] [5	6	۲						
78	SELECT '	* from	i bank_c	lata;						
Run	time Log	Re	rsult[1]							
	A			в			с		D	
1	age	~	job		<	marital		✓ ed	ucation	<
2	53		technician			married		un	known	
3	3 28		management			single		un	iversity.degree	
4	4 39		services			married		hiş	gh.school	
5	55		retired			married		ba	sic.4y	
6	6 30		managen	nent		divorced		ba	sic.4y	
7	37		blue-collar			married		ba	sic.4y	
8	8 39		blue-collar			divorced		ba	basic.9y	
9	36		admin.			married		un	iversity.degree	
10	10 27		blue-collar			single		ba	basic.4y	

- ・ Hidden column: 1 つ以上の非表示になっている列を選択し、列を非表示にします。
- Copy the row: コピーする 1 つ以上の行の左側を選択し、[Copy the row] をクリックします。
- · Copy the column: 上部の列で1つ以上の列またはポイントを選択し列をコピーします。
- · Copy: 選択したコンテンツを自由にコピーします。
- ・ Search: クエリ結果の右上隅に検索ボックスが表示されます。これを使ってテーブル内の データを簡単に検索できます。

5. ノードスケジューリングの設定

ノードタスク編集エリアの右側にある [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」をご参照ください。

6. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックの解除) します。

7. ノードタスクを発行します。

操作に関する詳細は、「リリースの管理」をご参照ください。

8. 本番環境でテストします。

操作に関する詳細は、「#unique_20」をご参照ください。

1.4.4 SQL コンポーネントノード

手順

 [Data Development] の [Business Flow] を右クリックし、[Create Business Flow] を選 択します。



2. [Data Development] を右クリックし、[Create Data Development Node] > [SQL Component Node] の順に選択します。

Ш	Data Developn 오 怘 [C C U	Sh test	Shell ×	Mr testMF	× ×	Ja te
0	Enter a file or creator name] [5]	60) :
*	> Solution			#!/bin/ #*****	bash ********	******	****
L M	 Business Flow Business Flow Base_cdp 			##autho ##creat	r:wangdan e time:20	18-09-0	3 00:0 ****
∎ V	> 😑 Data Integration > 🕢 Data Developm	n					
#	● Sq insert_dε ● [vī] start Me	Create Data Dev Create Folder	elopmentN	ode ID	ODPS SQL Shell		
	Mr testMR	Board Reference Comp	oonent		Virtual No PvODPS	de	
<u>₩</u>	> III Table				SQL Comp	oonent Noo	le
	> 🧭 Resource					SQL Co	mpone

- 開発の効率を向上させるため、データタスクの開発者はプロジェクトメンバーとテナントメン バーによって提供されるコンポーネントを使用して、データプロセスノードを作成することが できます。
 - ・ローカルプロジェクトのメンバーによって作成されたコンポーネントは、Project Componentsの下にあります。
 - テナントメンバーによって作成されたコンポーネントは、Public Componentsの下にあります。

ノードを作成する際は、ノードタイプを「SQL Component node」 タイプへ設定し、ノー ド名を指定します。

	Components	L, C,
(7) (7)	Project-specific	Create
*	None	
21		
Ē		

選択したコンポーネントのパラメーターを指定します。

ш	Tables	СС	🖆 myComponent 🌑	f testSQLComponent ×	testMR ×	🔙 testJAR.jar 🛛 🗙	Sq rpt_user_info_d x	See dw_user_info_all_d ×	🔄 ods_log_info_d 🗙	
			🖱 🙃 🖬	- 1 Q O	• 28					
*	Y 🛅 Tables						X			
R	✓ Conters						• Owner :	wangdan		
8	🗰 bank_data			t: <u>https://help.aliy</u>	un.com/document_	detail/30290.htm	Description :			
2	dw_user_info_all_d			erwrite table @@{my_	output_table}		Input Parameters(?)			
=	🗰 ods_log_info_d		9 partition 10 select 11 *	(ds=`\${blzdate}`)			* Parameter Name	mycompent	* Type : String	
5	edenavdogra ods_user_info_d VITONMENI result_table		12 from 13 @@{my 14 where c 15 AND s	_input_table} ategory in ('@@{my_i ubstr(pt, 1, 8) in (<pre>nput_parameter1} '\${bizdate}')</pre>	', '00{my_input_	: Description :	default value		
	m rpt_user_info_d						Default Value :	bank_data		
							Output Parameters()			
					不		* Parameter Name	4	* Type : String	
					57 57		Description :			
۵							Default Value :	4		

パラメーター名を入力し、パラメータータイプを「Table」または「String」に設定します。 3つの get_top_n パラメーターを順番に指定します。

Table タイプのパラメーターに対する入力テーブル"test_project.test_table'を指定します

4. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Scheduling Configuration] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」を ご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックの解除) します。

6. ノードを発行します。

操作の詳細については、「リリースの管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

SQL コンポーネントノードのバージョンのアップグレード

コンポーネント開発者が新しいバージョンをリリースした後、コンポーネントユーザーは、既存 のコンポーネントのインスタンスを最新版のコンポーネントへアップグレードするかどうか選択 することができます。

コンポーネントのバージョンメカニズムにより、開発者はコンポーネントを継続してアップグ レードすることができるため、アップグレード後ユーザーは向上したプロセス実行効率や最適化 されたビジネス効果を享受することができます。

たとえば、ユーザー A が ユーザー C が開発した v1.0 コンポーネントを使用しており、コンポー ネントオーナーの C はコンポーネントを V.2.0 ヘアップグレードします。 アップグレード後、 ユーザー A は v1.0 コンポーネントを継続して使用できますが、アップグレードリマインダーを 受け取ります。 旧コードと新コードを比較した後、ユーザー A が新しいバージョンのビジネス効 果の方が旧バージョンよりも良いと思った場合は、コンポーネントを最新版へアップグレードす るかどうかを判断することができます。

コンポーネントテンプレートに基づいて開発された SQL コンポーネントノードをアップグレード する場合、[Upgrade] を選択し、SQL コンポーネントノードが新しいバージョンでも有効かど うかパラメーターの設定で確認します。新しいバージョンのコンポーネントの指示に従って必要 な調整を行い、通常の SQL コンポーネントノードと同じようにノードを送信しリリースします。

インターフェイスの機能

ш	Components	C mycomponent x C tetsSQLComponent x (in testSHELL x (in testSHELL x (in testSHEL x (in testSHEL x (in testSHE	
-	Project-specific Public	□ ☆ J1 ⊕ ǎ Q ⊙ ● X	
*	A mycomponent wangdan 09-03 00:06:15	1 SQL component model	
R		3author:wangdan Besics 4create time:2018-09-03 00:06:14	
		5document: <u>https://help.aliyun.com/document_detail/30290.htm</u>	
		7 8 insert overwrite table @@(my_output_table) *Owner: wangdan	
		9 partition (ds-`\${bizdate}`) 10 select Description:	
R		12 from 12 from 13 A0/muttable\	
5		14 where category in ('@@(my_input_parameter1)', '@@(my_input_ 15 AND substrict, 1, 8) in ('\$(bizdate)')	
Ť		16 ; *Parameter Name *Type: String ~	
		Description :	
		T Default Value :	
		Cutput Parameters(7)	
		* Parameter Name * Type: String ~	
0			L

インターフェイスの機能は次のとおりです。

No.	機能	説明
1	Save	クリックして、現在のコンポーネントの設定を保存しま す。
2	Steal lock Edit	現在のコンポーネントのオーナーでない場合、クリック して、ロックを解除してノードを編集することができま す。
3	Submit	クリックして、現在のコンポーネントを開発環境へ送信 します。
4	Publish Component	クリックして、すべてのテナントに対しユニバーサルグ ローバルコンポーネントを発行します。これによりテナ ント内のすべてのユーザーがパブリックコメントを表示 し使用することができます。
5	Resolve Input and Output Parameters	クリックして、現在のコードの出入力パラメーターを解 決します。
6	Precompilation	クリックして、現在のコンポーネントのカスタムパラ メーターおよびコンポーネントパラメーターを編集しま す。
7	Run	クリックして、開発環境内でコンポーネントをローカル 上で実行します。
8	Stop Run	クリックして、実行中のコンポーネントを停止します。
9	Format	クリックして、キーワード別に現在のコンポーネント コードをソートします。

No.	機能	説明
10	Parameter Settings	クリックして、コンポーネントの情報、入力パラメー ターの設定、出力パラメーターの設定を表示します。
11	Version	クリックして、現在のコンポーネントの送信およびリ リースレコードを表示します。
12	Reference Records	クリックして、コンポーネントの使用レコードを表示し ます。

1.4.5 仮想ノード

仮想ノードは、データを生成しない制御ノードです。 通常、ワークフローのノードの全体的な計 画に対するルートノードとして使用されます。

三注:

ワークフローの最終出力テーブルには、複数の分岐入力テーブルが含まれます。 仮想ノードは 通常、これら入力テーブルに依存関係がない場合に使用されます。

仮想ノードタスクの作成

1. [Data Development] で [Business Flow] を右クリックし、[Create Business Flow] を選 択します。



[Data Development] を右クリックし、[Create Data Development Node] > [Virtual Node] の順に選択します。

	Data Developn 🖉 🛱 📮 Ċ 🕀) Lí
(7)	Enter a file or creator name	™
*	> Solution	
R	➤ Business Flow	
Ē	 A base_cdp Data Integration 	
# ≤ 0	 Opta Development ・ ・	Create Data DevelopmentNode ID > ODPS SQL Create Folder Shell Board ODPS MR
54 54	 Mr testMR Me锁定 09-02 Sh testSHELL Me锁定 09-02 S¹ testSHELL Me锁定 09-02 S¹ testSQLComponent Me 	Reference Component Virtual Node -03 000 PyODPS -03 000 SQL Component Node
Ť	> 🥅 Table	OPEN MR
	> 🙋 Resource > 🔂 Function	

3. ノードタイプを [Virtual Node] に設定し、ノード名を入力します。ターゲットフォルダーを 選択し、[Submit] をクリックします。

Create Node		×
Node Type :	Virtual Node ~	
Node Name :	testVirtual	
Destination Folder :		
	Submit	Cancel

4. ノードコードの編集: 仮想ノードのコードを編集する必要はありません。

5. ノードスケジューリングの設定

ノードタスク編集エリアの右側にある [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」 をご参照ください。

6. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、 または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックの解除) します。

7. ノードタスクを発行します。

操作の詳細については、「リリースの管理」をご参照ください。

8. 本番環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

1.4.6 ODPS MR ノード

MaxCompute では、MapReduce プログラミング API を使用できます。 MaxCompute でデータを処理するためには、MapReduce によって提供される Java API を使っ

て、**MapReduce** プログラムを記述します。 **ODPS MR** ノードを作成し、タスクスケジューリ ングで使用することができます。

DPS MR の編集方法や使用方法については、**MaxCompute** ドキュメントの「*WordCount* 例」に ある例をご参照ください。

ODPS MR ノードを使用するには、使用するリソースをアップロードし、リリースします。その 後、ODPS MR ノードを作成します。

リソースインスタンスの作成

1. [Data Development] の [Business Flow] を右クリックし、[Create Business Flow] を選 択します。

111	Data Developn 🖉 🗒 [🛱 Ċ 🕀 🕼 🚺 🖬 Data Developm
0	Enter a file or creator name	Solution New
*	> Solution	Business Flow New
民	✓ Business Flow	Folder Business Flow
e		Data Integration >
	> 品 原原体制1	Data Development 🔉
Ž	> 品 市田満見2	Table
#	> 🏯 西田市市山	Resource >
	> 品 教養的社会	Function
5	> 🛃 Bl_Demo	Function
f_{\times}	🗸 🛃 BirdLiu	
	N 📑 Data lata arati	

2. [Resource] を右クリックし、[Create Resource] > [jar] の順に選択します。

Create Resource		×	
* Resource Name :	testJAR.jar		
Destination Folder :			
Resource Type :	JAR ~		
	Upload to ODPS The resource will also be uploaded to ODPS.		
File :	Upload		
		Cancel	

3. 命名規則に従って、[Create Resource] にリソース名を入力します。リソースタイプを 「jar」に設定し、ローカルの jar パッケージを選択します。

Create Resource				×
* Resource Name :	test.JAR.jar			
Destination Folder :				
Resource Type :	JAR	*		
	Upload to ODPS The resource will also be uploaded to ODPS.			
File :	Upload			
		ОК	Cancel	

注:

- jar パッケージが ODPS クライアントへアップロードされた後、[Uploaded to ODPS] の 選択を解除します。解除をしないと、アップロードプロセス中にエラーが報告されます。
- ・ リソース名は、アップロードファイルと同じ名前にする必要はありません。
- リソース名の命名規則:1 文字以上 128 文字以内の文字列で、文字、数字、アンダースコア、ドットを使用できます。 リソース名は大文字と小文字が区別されません。 リソースが jar リソースの場合、拡張子は .jar です。 リソースが Python リソースの場合、拡張子は .py です。

4. [Submit] をクリックし、リソースを開発スケジューリングサーバーへ送信します。

Upload Resource	
Saved Files :	ip2region.jar
Unique Resource Identifier :	OSS-KEY-160u5o1g7t3g9uuim6j6polz
	Upload to ODPS The resource will also be uploaded to ODPS.
Re-upload :	Upload

5. ノードタスクを発行します。

操作の詳細については、「リリースの管理」をご参照ください。

ODPS MR ノードの作成

1. [Data Development] の [Business Flow] を右クリックし、[Create Business Flow] を選 択します。


2. [Data Development] を右クリックし、[Create Data Development Node] > [ODPS MR] の順に選択します。



3. ノードコードを編集します。 新規の **ODPS MR** ノードをダブルクリックし、次のインター フェイスへ移動します。

Data Developn 온 🗟 🕻 C 🕀 🗹	Ja ip2r	egion.jar ×	w test	MR 🖣	• •	数据开发		vi workshop_start x	<pre>sq create_table_ddl x</pre>	testMR
Enter a file or creator name	۳	B 0.	[5]		€					
> E Function		odps r	r		••••	•••••	••••		······································	
> 📙 Algorithm		author	time:2	018-09-	17 10	5:17:18				
> 🧾 control		*****	******	******	****	*******	****	*************	*****************	
> 🚣 works										
👻 🏯 workshop										
> 🔁 Data Integration										
Y 🚮 Data Development										
Sq create_table_ddl dataworka_										
• Mr testMR Melocked 09-171										
Sq dw_user_info_all_d datawork										
a ods_log_info_d dataworks_d										
Sq rpt_user_info_d Mellocked 0										
vi workshop_start Melocked 0										
> 🧾 Table										
✓ Z Resource										
ip2region.jar Mellocked 09-1										
• 🌆 testJARjar dataworks_demo										

ノードコードの編集例:

jar -resources base_test.jar -classpath ./base_test.jar com.taobao. edp.odps.brandnormalize.Word.NormalizeWordAll

コードの説明は次のとおりです。

- ・ -resources base_test.jar:参照される jar リソースのファイル名を示します。
- ・ -classpath: jar パッケージのパスです。
- com.taobao.edp.odps.brandnormalize.Word.NormalizeWordAll: 実行中に呼び 出される jar パッケージのメインクラスを示します。 jar パッケージのメインクラス名と 同じにする必要があります。

1 つの MR で複数の jar リソースを呼び出す場合、クラスパスは-classpath ./xxxx1. jar,./xxxx2.jar のように記述します。2 つのパスは、コンマで区切ります。

4. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」 をご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押して ノードを開発環境へ送信 (およびロックの解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリースの管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

1.4.7 SHELL ノード

SHELL タスクは、標準の SHELL 構文をサポートしていますが、対話型の構文はサポートして いません。 SHELL タスクは、デフォルトのリソースグループで実行することができます。 IP ア ドレスやドメイン名へアクセスする場合、[Project Configuration] を選択し、IP アドレスま たはドメイン名をホワイトリストに追加します。

手順

1. [Data Development] の [Business Flow] を右クリックし、[Create Business Flow] を選 択します。



2. [Data Development] を右クリックし、[Create Data Development Node] > [SHELL] の 順に選択します。

111	Data Developn 🖉 🛱 📮	ц	Mr test	MR	×	Ja test.	JAR.jar	×	Sq rpt	
0)	Enter a file or creator name		V.		₿	[↑]	[১]		C) :
*	> Solution				odj	os m ∗∗∗∗	r ******	*****	***	*****
R	➤ Business Flow				aut	thor	:wangda	in		
ė.	✓ ♣ base_cdp				cre	eate ****	time:2	2018-0	9-0 ***	2 23:5 *****
×	> 🧧 Data Integration > 🕢 Data Developmen									
Ħ	● Sq insert_data	Create D	ata Dev	velopmentNode ID >			ODPS SQL			
_	• vi start Me钳	Create F	older							
E.	• Mr testMR M	e Com	Nonent Virtual No			Shell				
∱×	> 🧮 Table	Je com	ponent			PyOD	PS			
	> 🧭 Resource						SQL C	ompone	nt N	ode
Ш	> 📴 Function						OPEN	MR		

- 3. ノードタイプを「SHELL」に設定し、ノード名を入力します。ターゲットフォルダを選択 し、[Submit] をクリックします。
- 4. ノードコードを編集します。
 - SHELL ノードコード編集ページへ移動し、コードを編集します。

Enter a file or creator name Image: Image: Enter a file or creator name Image: Image		Data Developn 🖉 🗟 📮 C 🕀 🕁	Sh testS	SHELL ×	Mr testMR	× 🌆 👻	The life has been created.	Sq dw_user_info_all
* Solution Image: solution ##/bin/bash * Business Flow Image: solution ##author:wangdan * Solution ##author:wangdan * Solution ##create time:2018-09-03 00:02:13 * Op Data Integration ************************************	673	Enter a file or creator name		🖺 🔿	6			
× Business Flow × Business Flow <td< th=""><th>*</th><th>> Solution</th><th></th><th>#!/bin/b</th><th>ash</th><th>*********</th><th>***********************</th><th>************</th></td<>	*	> Solution		#!/bin/b	ash	*********	***********************	************
• ▲ base_cdp • ▲ ##create time:2018-09-03 00:02:13 #*************************	R	➤ Business Flow		##author				
→ Data Integration 6 ▲ ✓ OD Data Development	в	🗙 🛃 base_cdp		##create	time:2018-0	09-03 00:02		.**************
🖻 🗸 🕐 Data Development		> 📴 Data Integration						
		✓ ☑ Data Development						
● 函 insert_data Me帧定 08-31 15:	ŧ	• 🛯 insert_data Me锁定 08-31 15						

SHELL 文でシステムスケジューリングパラメーターを呼び出す場合、SHELL 文を次のとお りコンパイルします。

echo "\$1 \$2 \$3"		
(2) 注:		

パラメーター1パラメーター2... 複数のパラメーターはスペースで区切ります。 システムの スケジューリングパラメーターの使用方法に関する詳細は、「パラメーターの設定」をご参 照ください。

5. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Scheduling Configuration] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」を ご参照ください。

6. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックの解除) します。

7. ノードタスクをリリースします。

操作の詳細については、「リリースの管理」をご参照ください。

8. 本番環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

使用例

SHELL を使ってデータベースへ接続します。

 Alibaba Cloud でデータベースを作成し、リージョンが中国 (上海)の場合、データベースへ 接続するには、次のホワイトリスト内の IP アドレスでデータベースを開きます。

10.152.69.0/24、10.153.136.0/24、10.143.32.0/24、120.27.160.26、10.46.67.156、120.27.160.81,10.46.64.81、121.43.110.160、10.117.39.238、121.43.112.137、10.117.28.203、118.178.84.74、10.27.63.41、118.178.56.228,10.27.63.60、118.178.59.233、10.27.63.38、118.178.142.154、10.27.63.15、100.64.0.0/8

注:

Alibaba Cloud でデータベースを作成したが、リージョンが中国 (上海) ではない場合、イ ンターネットを使用するか、またはデータベースと同じリージョンで ECS インスタンスをス ケジューリングリソースとして購入し、カスタムリソースグループで SHELL タスクを実行 することを推奨します。

・データベースをローカルで作成した場合は、インターネットを使って上記のホワイトリスト内のIPアドレスでデータベースを開くことを推奨します。



カスタムリソースグループを使って SHELL タスクを実行する場合、カスタムリソースグ ループマシンの IP アドレスを上記のホワイトリストへ追加する必要があります。

1.4.8 PyODPS ノード

DataWorks では、PyODPS タスクタイプを提供し MaxCompute の Python SDK を統 合することもできます。 Python コードを直接編集し、DataWorks の PyODPS ノードで MaxCompute を操作することができます。

MaxCompute では、**MaxCompute** を操作するために使用することができる *Python SDK* が提 供されます。

🗎 注:

Python 2.7 は基礎となるレイヤーで使用されます。 PyODPS ノードプロセスのデータサイズ は、50MB を超えることはできません。使用するメモリ量は 1GB を超えることができません。

PyODPS ノードの作成

1. [Data Development] で [Business Flow] を右クリックし、[Create Business Flow] を選 択します。



2. [Data Development] を右クリックし、[Create Data Development Node] > [PyODPS] を選択します。

111	Data Developn 🖉 🛱 📮	С 🕀 Ф	Mr testMR	× Ja testJAR.jar × Sq rpt
(/)	Enter a file or creator name	V.		1 b 🗇 🕑 i
*	> Solution		1od	ps mr *****************************
R	 Business Flow 		- 3au	thor:wangdan
Ē	✓ 🛃 base_cdp		4 cr 5**	eate time:2018-09-02 23:5
-	Data Integration			
	🗸 🚺 Data Developme	nt		
	● Sq insert_data	Create Data De	velopmentNode ID	> ODPS SQL
#	● 🔽 start Me∜	Create Folder		Shell
R	■ Mr testMR M	Board		ODPS MR
		Reference Com	ponent	Virtual Node
f_{\times}	> 🧾 Table			PyODPS
5	> 🧭 Resource			SQL Component Node
Ħ	> 🔁 Function			OPEN MR
	> 📒 Algorithm			

- 3. PyODPS ノードを編集します。
 - a. ODPS ポータル

DataWorks では、PyODPS ノードはグローバル変数 ODPS または ODPS エントリであ る o が含まれています。手動で ODPS エントリを定義する必要がありません。

```
print(odps.exist_table('PyODPS_iris'))
```

b. SQL 文を実行します。

PyODPS では、**ODPS SQL** クエリをサポートし、実行結果を読むことができます。 **execute_sql** または **run_sql** メソッドのリターン値は実行中インスタンスです。

ODPS コンソールで実行できるコマンドのすべてが **ODPS** で受け入れられる **SQL** 文では ありません。 非 **DDL/DML** 文を呼び出すには、他の方法を使用する必要があります。 た とえば、**run_security_query** メソッドを使って **GRANT** 文または **REVOKE** 文を呼び 出します。**run_xflow** または **execute_xflow** メソッドを使って **PAI** コマンドを呼び出 します。

o.execute_sql('select * from dual') # Run the SQL statements in synchronous mode. Blocking continues until execution of the SQL statement is completed. instance = o.runsql('select * from dual') # Run the SQL statements in asynchronous mode. print(instance.getlogview_address()) # Obtain the logview address . instance.waitforsuccess() # Blocking continues until execution of the SQL statement is completed.

c. ランタイムパラメーターを設定します。

ランタイムパラメーターは必ず設定する必要があります。 パラメータータイプ dict を使っ てヒントパラメーターを設定することもできます。

o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper .split.size': 16})

sql. 設定をグローバル設定へ追加した後、関連のランタイムパラメーターがそれぞれの実行中の python へ追加されます。

from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris') # "hints" is added
based on the global configuration.

d. SQL 文の実行結果を読み取ります。

SQL 文を実行するインスタンスは、open_reader 操作を直接行うことができます。 この 場合、構造化されたデータは SQL 文の実行結果として返されます。

with o.execute_sql('select * from dual').open_reader() as reader: for record in reader: # Process each record.

別の場合では、desc が SQL 文で実行される場合があります。 この場合、オリジナルの

SQL 文の実行結果は、reader.raw 属性から取得されます。

with o.execute_sql('desc dual').open_reader() as reader: print(reader.raw)

🗎 注:

ユーザーが定義したスケジューリングパラメーターは、データ開発に使用されます。 PyODPS ノードがページ上で直接トリガーされる場合、時間は明確に指定する必要があり ます。 **PyODPS** ノードの時間は、**SQL** ノードの時間のように直接置き換えることができ ません。

次のようにシステムパラメーターを設定することができます。

Py Pyte	est	×	Fx upp	erlower_	java x	Ja (upper.jar	×	Di lo	ghub	×
	[↑]	[b]	P	谢	Þ		С	\leq		23	:
1	a =	'\${b	dp.syst	em.bi	.zdate}	·'					
2	prin	t (f	ormat(a	a))							
3	b =	'\${b	dp.syst	.cm.cy	<pre>vctime}</pre>	'					
4	prin	t (fo	ormat()) [¯]							
	1		Ì	· · ·							

次のようにユーザー定義パラメーターを設定することができます。

	O&M
1 print (args['ds']) X Basics @	Schedne
Node Name: Pytest Node ID: 700001928004	Ne Ne
Node Type: PyODPS Owner: dtplus_docs ~	lations
Description: datetest	
Parameters: ds=\${yyymmdd-1}	versio

4. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Schedule] をクリックし、[node scheduling

configuration] ページへ移動します。詳細は、「スケジューリングの設定」 をご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックする、または ctrl + S を押して ノードを開発環境へ送信 (およびロックの解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリースの管理」をご参照ください。

7. 運用環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

1.4.9 クロステナントノード

このトピックでは、異なるテナントのノードを関連付けるために通常使用されるクロステナント ノードについて説明します。 クロステナントノードは送信側ノードと受信側ノードに分けられま す。

前提条件

送信側ノードと受信側ノードは同じ Cron 式を使用する必要があります。 [Schedule] > [Scheduling Mode]を選択して、次の図に示すように、Cron 式を表示します。

×		Sch
		edule
Scheduling Mode ⑦		
Instance Created :	Next Day Immediately After Publishing Note: Dependencies configured will not take effect immediately after publishing.	
Schedule :	Normal ○ Zero-load	
An error occurred. Try again. :	0	
Effective Period :	1970-01-01 🗎	
Pause Scheduling :	Note. The schedule runs only in the effective period.	
D		
Recurrence .		
Specify Time :		
Run At :	00:22 ③	
CRON Expression :	00 22 00 ** ?	
Depend on Last Interval :		

ノードの作成

1. [Data Studio]ページで、[Control] を右クリックし、 [Create Control Node] > [Cross-Tenant Node] を選択します。



ダイアログボックスに名前を入力して[Submit]をクリックします。

2. ノード設定を完了します。 ノードタイプを[Send]または[Receive]に設定します。 対象の ワークスペースと Alibaba Cloud アカウントを承認します。 この例では、ノードタイプを Send に設定します。 したがって、受信側ノードによって承認されたワークスペースとアカウ ントを入力する必要があります。 ノード設定が完了したら、ノードを保存して送信します。

C								
Cross-Tenant Node								
Node Identifier : dtplus_doc								
r an account.								
Enter the workspace name.								
Workspace	Actions							
Ofghus.00C	Delete							
	C The DOC' Second to the Workspace name.	C The continue of the set of the						

同じ手順に従って、受信側のアカウントとワークスペースの下にコントロールノードを作成し ます。ノードタイプを受信に設定します。その後、利用可能な送信側ノードに関する情報が 表示されます。 タイムアウトタイマーも設定する必要があります。 受信側ノードの実行が開 始されると、タイムアウトタイマーが再起動します。

E F & C	
Cross-Tenant Node	
Cross-Tenant Nodes Available to Receive :	Refresh
 dataworkadataka? workshop_marka 	
tesking The D	
Timeout : 30 min	

送信側ノードはメッセージをメッセージセンターに送信し、メッセージが正常に配信された後 でメッセージの実行を開始します。 受信側ノードは、メッセージセンターから継続的にメッ セージを引き出します。 受信側ノードは、タイムアウト期間内にメッセージを正常にプルした ときに実行を開始します。

受信側ノードがタイムアウト期間内にメッセージを受信しないときは、タスクは失敗します。 メッセージのタイムアウトは最大**24**時間に設定できます。

例:

2018年10月8日、定期的に作成されたインスタンスが正常に実行され、メッセージがメッ セージセンターに送信されました。2018年10月7日に設定された営業日で受信側ノードの遡 及インスタンスを作成すると、受信者ノードが表示されます。

1.4.10 マージノード (Merge node)

本ページでは、マージノードの概念と作成方法、およびマージロジックの定義方法について説明 します。 実例を使ったマージノードのスケジューリング設定や操作の詳細についても説明しま す。

概念

・マージノードとは、DataStudioが提供する論理制御ファミリーノードの1つです。

- ・依存関係のマウントや分岐ノードの下位ノードの実行トリガーに関する問題を解決するため
 に、マージノードでは、上位ノードの実行状態を統合できます。
- ・現在のマージノードの論理定義は、ノードの実行状態の選択をサポートしていませんが、分岐 ノードの複数の下位ノードを正常に統合することをサポートします。下位ノードは、依存関係 としてマージノードを直接マウントすることができるようになります。

たとえば、分岐ノード C が論理的な排他的分岐 C1 と C2 の 2 つを定義します。 異なる分岐で は、異なるロジックを使用して、同じMaxCompute テーブルへ書き込みされます。 下位ノード B がこのMaxCompute テーブルの出力に依存する場合、マージノード J を使って分岐を統合し た後にマージノード J を B の上位の依存関係へ追加します。B が C1 と C2 に直接マウントされ る場合、いつでも、C1 と C2 のどちらかが必ず失敗します。これは、満たされない分岐条件のた めであり、スケジュールでは B の実行をトリガーできないためです。

マージノードの作成

[Merge Node] は、新しいノードメニューの [Control] クラスディレクトリにあります。

DataV	DataStudio	MexCompute_DOC	~				
111	Data Development	₽ B C C O	Ŀ	Å 5440	niole.1219	m1 ×	
(J)	Enter a file or creator nam	Solution New			<u>ئ</u>		С
*	> Solution	Business Flow New					
Q	> Business Flow	Folder		B	ranch Logio	c defini	tion
6	> Workflow (Early Version	Data Integration	>		Add Branch	1	
č		Data Development	>		Branch		
×		Table					
≡		Resource	>				
∎		Function					
fx		Algorithm					
_		Control	>	Cross-ter	nant node		
				The OSS	object Inspe	ction	
Σ				Assignm	ent Node		
亩				Branch n	ode		
				Merge No	ode		

マージロジックの定義

マージ分岐を追加します。出力名、または親ノードの出力テーブル名を入力します。[add] をク リックし、マージ状態のレコードを確認します。実行結果では、実行状態が表示されます。現在 は、次の図に示すとおり2つの状態 (Successful、Branch not running) のみがサポートされ ています。

Sq Branch_2 × Sq Branch_	1 × 🛃 Workflow_migration × 🦞 Merge_node_	121901 ×		
	ž			
Merge Logic definition				
Add merge Branch: Enter an outp	but name or output table name 🛛 🖌 🕇			
Merge conditions is set				
Upstrear	n Node: MaxCompute_DOC.Branch_1	Operation State is equal :		
And V Upstream	n Node: MaxCompute_DOC.Branch_2	Operation State is equal :		
Implementation reculte is est				
implementation results is set				
·				
Is set this node operation state:				
Succeeded				

マージノードのスケジューリング属性は次の図に示すとおりです。

Dependencies ⑦ Auto Parse: • Yes \ No Parse I/0						Schedule
Upstream Node Enter an output name or output ta						Relation
Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID	Owner		ship
MaxCompute_DOC.Branch_1		Branch_1		tina	Added by Default	
MaxCompute_DOC.Branch_2		Branch_2		tina (Added by Default	
MaxCompute_DOC.500153431_out		Branch_1		tre	Added Manually	
MaxCompute_DOC.500153432_out		Branch_2		tina.	Added Manually	

マージノードの例

下位ノードでは、分岐ノードを上位ノードとして追加した後、対応する分岐ノード出力を選択す ることで、異なる条件にある分岐の方向を定義することができます。たとえば、次の図に示すビ ジネスプロセスの場合、"Branch_1"と "Branch_2" はどちらも分岐ノードの下位ノードです。



Branch_1 は、次の図に示すとおり、autotest.fenzhi121902_1 の出力に依存します。

Sq Brar	🔄 Branch_2 x 🔄 Branch_1 x 👗 Workflow_migration x 🦞 Merge_node_121901 x							
e								
1 2 3		×						
4 5 6 7		Dependencies ⑦ Auto Parse: ⑦ Yes ○ No Parse I/0 Upstream Node Enter an output name or output table name ∨ + Use The Workspace Root Node						
		Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID	Owner		
		autotest.fenzhi121902_1		Branch_node_121902		tne	Added Manually	

Branch_2は、次の図に示すとおり、autotest.fenzhi121902_2の出力に依存します。

Sq Bran	nch_2 × Sq Branch_1	× 🛃 Workflow_migration × 🖞 Mer	ge_node_121901 ×					
Ľ								
1 2		×						
3		Dependencies ⑦	Dependencies (7)					
4 5		Auto Parse: No Parse I/O						
6		Upstream Node Enter an output name o	Upstream Node Enter an output name or output table name > + Use The Workspace Root Node					
		Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID			
		autotest.fenzhi121902_2		Branch_node_121902		tra	Added Manually	

マージノードのスケジューリング属性は次の図に示すとおりです。

Sq Branch_2	× Sq Branch_1 × 🎄 Workflow_migration	× Y Merge_node_121901 ×						
<u> </u>	E 🕆 C							
Merge Logi	c definition		×					
Add merge Brar	Ich: Enter an output name or output table name		Dependencies ⑦					
			Auto Parse : 💿 Yes 🔿 No					
Merge condition	is is set		Upstream Node Enter an out					
	Upstream MaxCompute_DOC.Branch_1 Node :	Operation State Succeeded ×	Upstream Node Output Name					
And 🗸	Upstream MaxCompute_DOC.Branch_2	Operation State Succeeded ×	MaxCompute_DOC.Branch_1		Branch_1	tine	Added by Default	
	Node :	is equal : Branch is not running ×	MaxCompute_DOC.Branch_2		Branch_2	tina	Added by Default	
Implementation	raeulte ie ear		MaxCompute_DOC.50015343	11_out -	Branch_1	tine	Added Manually	
In port this ports	enaration state:		MaxCompute_DOC.5001534	12_out -	Branch_2	tina	Added Manually	
Succeeded	operation state.		Output Enter on output on the					

タスクを実行します。

分岐条件が満たされている場合、実行する分岐の下位ノードを選択します。 実行の詳細は [Running Log] で確認できます。

分岐条件が満たされていない場合、実行する分岐の下位ノードは選択しません。 [Running Log] ではノードが "skip" に設定されていることが確認できます。

マージノードの下位ノードは正常に実行されています。

1.4.11 分岐ノード (Branch node)

分岐ノードは、**DataStudio**が提供する論理制御ファミリーノードの一つです。 分岐ノードは分 岐ロジックと異なる論理条件にある下位分岐の方向を定義することができます。

分岐ノードの作成

次の図に示すとおり、[Branch node] は、新しいノードメニューの [Control] クラスディレク トリにあります。

Data	DataStudio	MacCompute_DOC	~	
111	Data Development	₽₿₽°0€	⊡	🚠 Enerstuneds 121901 ×
ŝ	Enter a file or creator nam	Solution New		🗉 🗗 🗄 🗘
*	> Solution	Business Flow New		
Q	> Business Flow	Folder		Branch Logic definition
6	 Workflow (Early Version 	Data Integration	>	Add Branch
		Data Development	>	Branch
×		Table		
▦		Resource	>	
I		Function		
fy		Algorithm		
1^		Control	>	Cross-tenant node
				The OSS object Inspection
Σ				Assignment Node
亩				Branch node
				Merge Node

分岐ロジックの定義

1. 分岐ノードを作成した後、次の図に示すとおり [Branch Logic definition] ページへ移動し

ます。

👗 Branc	h_node_121901 ×					
BI	ranch Logic definition ⑦					
	Branch	Conditions	Associated to node output		Branch describe	Actions

 [Branch Logic definition] ページで、[Add Branch] ボタンを使って次の図に示すとおり [Branch Conditions]、[Associated to node output]、および[Branch describe] を定 義します。

Configuration branch Definition	×
Branch Conditions :	
Associated to node output :	
Branch describe :	
	Ok Cancel

パラメーターは次のとおりです。

- Branch Conditions
 - 分岐条件は、**Python** 比較演算子に基づいて論理判断条件の定義のみをサポートします。
 - 実行状態の式の値が「true」の場合、対応する分岐条件は満たされていることを意味し ます。「true」でない場合は、条件が満たされていません。
 - 実行状態の式に解析エラーが報告される場合は、分岐ノード全体の実行状態が失敗する ように設定されています。
 - 分岐条件は、ノードコンテキストで定義されているグローバル変数とパラメーターの使用をサポートします。図にある \${Input}のように、分岐ノードで定義されたノード入力パラメーターとすることができます。
- Associated to node output
 - ノード出力は、分岐ノードの下位ノードの依存関係をマウントするために使用します。
 - 分岐条件を満たす場合、関連するノード出力にマウントされる下位ノードは、実行するために選択されます(依存関係のあるその他の上位ノードのステータスもご参照ください)。
 - 分岐条件が満たされない場合、関連するノード出力にマウントされる下位ノードは、実行するためには選択されません。分岐条件を満たしていないため、下位ノードは実行中ではないステータスになります。
- ・ Branch describe: 分岐定義の説明をご参照ください。

2つの分岐の定義:次の図に示すとおり、\${Input}==1と\${Input}>2です。

👗 Brar	nch_node_121902 ×						
면							
	Branch Logic definition	0					
	Branch	Conditions	Associated to node output	Branch describe			
		S{input}==1	autotest.fenzhi121902_1		Edit Delete		
		\${input}>2	autotest.fenzhi121902_2				

- Edit: [Edit] ボタンをクリックします。分岐の設定を編集したり、関連の依存関係も変 更できます。
- Delete: [Delete] ボタンをクリックします。分岐の設定を削除したり、関連の依存関係 も変更できます。

Scheduling configuration

分岐条件を定義した後、[Schedule] のノード [Output] へ出力名が自動的に追加されます。下 位ノードはマウントする出力名に依存します。 次の図に示すとおりです。

Dependencies ⑦ Auto Parse: ③ Yes ○ No Parse I/0 Upstream Node Enter an output name or output	t table name 🗸 🕂 Use The V	Vorkspace	e Root Node					Schedule Relations
Upstream Node Output Name	Upstream Node Output Table Name		Node Name	Upstream Node ID	Owner			
MaxCompute_DOC 500153409_out			Assign_node_121902		wa	Added Manually		
Output Enter an output name	Output Enter an output name							
Output Name	Output Table Name	Dow	nstream Node Name	Downstream Node ID	Owner		Actions	
autotest.fenzhi121902_1	- Ø	Bran	ch_1		tra	Added by Default		
autotest.fenzhi121902_2	- Ø	Bran	ch_2		teo -	Added by Default		
MaxCompute_DOC.500153095_out	- @					Added by Default		
MaxCompute_DOC.Branch_node_121902	- Ø					Added Manually		

書き込みで確立されたコンテキスト依存関係に対するスケジューリング設定に出力レコードがな い場合は、手動で入力します。

出力ケース - 分岐ノードにマウントされる下位ノード

下位ノードでは、分岐ノードを上位ノードとして追加した後、対応する分岐ノードの出力を選択 することで、異なる条件の分岐の方向を定義することができます。たとえば、次の図に示すビジ ネスプロセスでは、**"Branch_1"**と **"Branch_2"** はどちらも分岐ノードの下位ノードです。



Branch_1 は、次の図に示すとおり、autotest.fenzhi121902_1 の出力に依存します。

₩ Ме	Y Merge_node_121901 x 🐼 Branch_2 x 🐼 Branch_1 x 👗 Workflow_migration x 🄄 Assign_node_121902 I at the state of the stat							
۳								
1 2 3		Nanandanaias @						
4 5 6		Auto Parse : • Yes O No Parse //						
7	SHOW tables;	Upstream Node Enter an output name						
		Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID			
		autotest.fenzhi121902_1		Branch_node_121902		are	Added Manually	
		Output Enter an output name						

Branch_2は、次の図に示すとおり、autotest.fenzhi121902_2の出力に依存します。

Sq Brar	nch_2 × Sq Branch_1	× 🛃 Workflow_migration × 🚑 Ass	ign_node_121902 k Branch_node_121902	× \$\$ Merge_node_121901 ×				
Ľ								
1 2		×						
3 4 5		Dependencies ⑦ Auto Parse : • Yes · No Parse I/0						
6		Upstream Node Enter an output name or output table name v + Use The Workspace Root Node						
		Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID	Owner		
		autotest.fenzhi121902_2		Branch_node_121902		tita	Added Manually	

スケジューリング操作の送信

オペレーションセンターへ実行するディスパッチを送信します。分岐ノードが条件を満たします (autotest.fenzhi121902_1 に依存します)。したがって、ログの印刷結果は次のとおりとなり ます。

- ・ 分岐条件が満たされている場合、実行する分岐の下位ノードを選択します。 実行の詳細は [Running Log] で確認できます。
- ・ 分岐条件が満たされていない場合、実行する分岐の下位ノードは選択しません。 [Running Log] ではノードが "skip" に設定されていることが確認できます。

追記: サポートされている Python 比較演算子

次の表では、変数 a は10、変数 b は 20 を前提にしています。

比較演算子	説明	例
==	イコール - 等値のオブジェクトを比 較します。	(a==b) は 'false' を返します。
!=	イコールではない - 等値ではない 2 つのオブジェクトを比較します。	(a!=b) は 'true' を返します。
<>	イコールではない - 等値ではない 2 つのオブジェクトを比較します。	(a<>b) は 'true' を返します。 この 演算子は、'!=' に似ています。
>	より大きい - x が y より大きいかど うかを返します。	(a>b) は 'false' を返します。
<	より小さい - x が y より小さいかど うかを返します。すべての比較演算 子は、true の場合は 1、false の 場合は 2 を返します。 特別な変数 True および False と同等です。	(a <b) 'true'="" td="" は="" を返します。<=""></b)>
>=	以上 または イコール - x が y より も大きい、または y と等値である かどうかを返します。	(a>=b) は 'false' を返します。
<=	以下またはイコール - x が y よりも 小さい、または y と等値であるか どうかを返します。	(a<=b) は 'true' を返します。

1.4.12 割り当てノード (Assignment node)

割り当てノード (Assignment node) は特別なタイプのノードです。 ノードのダウンストリー ムで参照しそれらの値を使用するために、ノードでコードを書き込むことによって出力パラメー ターの割り当てをサポートし、ノードのコンテキストを使って組み合わせて転送します。

割り当てノードの作成

次の図に示すとおり、新しいノードメニューの [Control] クラスディレクトリに

は、[Assignment Node] が配置されています。



割り当てノードの値ロジックの記述

割り当てノードには、[Node Context] の出力に名前をつける、固定出力パラメーターがありま す。 MaxCompute、Shell、Python の使用をサポートし、パラメーターを割り当てるコード を書き込みます。これらの値はノードコードの操作および計算結果です。1つの割り当てコード に対して選択できる言語は1つです。

	٤.		С		
	Please	select a	ssignment langua	ge: ODPS SQL	Please select assignment language type this options in node submitted after don't allow modify.
				ODPS SQL	
				SHELL	
				Python	

📔 注:

・ 出力パラメーターの値は、次のように、コードの最終出力行から取得されます。

- MaxCompute SQL の最終行にある SELECT 文の出力
- shell の最終行の ECHO 文からのデータ
- Python の最終行の PRINT 文の出力
- ・出力パラメーター値には制限があり、最大転送値は 2M です。割り当て文の出力がこの制限 を超える場合、割り当てノードは実行に失敗します。

The Node Output Parameters Add								
No.	Parameter Name	Туре	Value	Description	Source	Actions		
1	outputs	Variable	\${outputs}	Nontralaises , Breatlik/Tech.or	Added by Default			

ダウンストリームノードには割り当てノードの出力を使用します。

ダウンストリームノードでは、割り当てノードをアップストリーム依存関係として追加した後、 ノードコンテキストを使って割り当てノードの出力をノードの入力パラメーターとして定義しま す。コードでそれを参照します。アップストリーム割り当てノードの出力パラメーターの特定値 が取得されます。詳細については、「ノードのコンテキスト」をご参照ください。

Node C	Node Context ③								
The Node	Input Parameters	Add							
No.	Parameter Name	Value Of The Source	Description	Parent Node ID	Source	Actions			
1	input	MarCompany_DOC.213:outputs	NET:2004年,NEG27时从2		Added by Default				

割り当てノードの例

1. ビジネスプロセスを作成し、次の図に示すとおり、以下のノードをそれぞれ作成します。



割り当てノードを設定する際、システムにはデフォルトで [outputs] パラメーターが表示されます。 実行後、関連のパラメーター結果が[Operation Center] > [Properties] > [Context] ページに表示されます。

S DataStudio MusiCompu	al.000 ~			Cro	ss-project cloning	Operation O	Center 🔍 🖷	ra Engl
🔲 Data De 온 🗟 📮 C 🕀 🖆	Assign_shell_1130 ×							
Enter a file or creator name								08/
Solution III	Please select assignment language SHELL ^	×						
Q × Business Flow Y & Workflow_migration O O O Data Integration	assignment language type this optic 00DPS SQL 2 SHELL	MaxCompute_DOC.shell_1129		shell_1129_c opy		tina Adde ally	d Manu 🔒	
Data Development	3 echo \$1; 4 5 echo 'this is name,ok';	Output Enter an output name						
> 🖉 Resource			Output Table Na me	Downstream Node N ame	Downstream Nod e ID	Own Sourc		
fx > E Algorithm		MaxCompute_DOC.500152852_out	- Ø			- Adder - ault	1 by Def 👘	
		MaxCompute_DOC.Assign_Shell_11 30 C	- Ø	Assign_shell_1131		in Adder	1 Manual 🔒	
		Node Context ⑦ The Node Input Parameters Add						
	不	The Node Output Parameters Add						
Ø.	кл КУ	1 outputs Var	iable \${outputs}	Marth Index Line . Th	Bess Finder	Added by Default		

3. 次の図に示すとおり、アップストリーム [outputs] パラメーターをダウンストリーム入力パラ メーターとして使用します。

Deta	DataStudio Muscompu	as_DOC ~					Cross-proje	ect cloning	Operation Center	A 100	Englist
ш	Data De 🙎 🛱 📮 C 🕀 🗹	🐣 Assign_shell_1131 🗴 💽 shell_1	131 🌔 🚠 Vilorichaw_migration 🗴 🚷 Assign_s								
Ø											08M
*	> Solution		×								
a	✓ Business Flow										
	🗸 🏯 Workflow_migration										
G	> 🚞 Data Integration	6	Marchingute DOCAssion_Shell_1131			Assign_shell_113	1	titta	Added Manually		
	> 🕢 Data Development	<pre>7 echo \${input[0]};</pre>									
	> 🧮 Table	• 9 echo 'hgagsgahg';									nsuit
	> 🙋 Resource		Enter an output name								
1	Function										
fx	> 🔚 Algorithm		New York and Add Intelligence and	<i>(a</i>)					And and have Destroyed		
_	🗸 🧭 Control		Haarcompute_pot_500+52926_out	- 6					Added by Default		
	 Assign_python_1130 		MarCarna et 1925 shell 1131	- 0					Added Manually		
Σ	Assign_shell_1130 Me										
亩	Assign_shell_1131										
	Assign_sqL1130 Mel		Node Context ②								
	s 🚠 test prijori										
	> 👗 volcing		The Node Input Parameters Add								
	 Workflow (Early Version) 										
			1 Input MaxCompute	e_DOC Assign_Shell_1131:outp	ts Militi	0646.0846	1793.952	Added b	y Default Edit Delet		
			The Node Output Parameters Add								

割り当てノードタスクを実行します。

兰注:

通常の操作やメンテナンスでは、上記の設定パラメーターはパッチデータ操作で検証することが できますが、テスト操作パラメーターは検証できません。

- 1. タスクが設定されスケジュールされると、通常、実行インスタンスが次の日に生成されます。
- ランタイムでは、コンテキストの入出力パラメーターを表示することができます。次のリンク をクリックして入出力結果を確認します。
- 3. [Running Log] では、"finalResult" からのコードの最終出力を確認できます。

. #************************************
echo \$1;
echo 'this is name,ok';
echo 'this is password';
shell output: shell
shell output: this is name,ok
(shell output: this is password)
2018-12-19 17:12:25:897 [main] INFO c.a.d.a.w.handler.AssignmentHandler
2018-12-19 17:12:26.897 [main] INFO c.a.d.a.w.handler.AssignmentHandler - result: this is password
2018-12-19 17:12:26.925 [main] INFO c.a.d.a.w.handler.AssignmentHandler - ===>{finalResult:) [["this is password"]]
2018-12-19 17:12:27.363 [main] INFO c.a.d.a.w.handler.AssignmentHandler - cost Time: 1
2018-12-19 17:12:27.363 [main] INFO c.a.dw.alisa.wrapper.ControllerWrapper - job finished!
2018-12-19 17:12:27,363 [Thread-2] INFO s.c.a.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@48cf768c: startup da
te [Wed Dec 19 17:12:24 CST 2018]; root of context hierarchy
2018-12-19 17:12:27.365 [Thread-2] INFO o.s.j.e.a.AnnotationMBeanExporter - Unregistering JMX-exposed beans on shutdown
2018-12-19 17:12:27 INFO
2018-12-19 17:12:27 INFO Exit code of the Shell command 0
2018-12-19 17:12:27 INFO Invocation of Shell command completed
2018-12-19 17:12:27 INFO Shell run successfully!
2018-12-19 17:12:27 INFO Current task status: FINISH
2018-12-19 17:12:27 INFO Cost time is: 4.131s
./home/admin/allantaskrede/taskinfo//38188218/phoenixprod/17/13/13/iogaras54/31850674ejurgin/13_1629174701.log-END-EOF

1.5 スケジューリングの設定

1.5.1 基本属性

下図は、基本属性を設定するためのインターフェイスです。

Basics ②						
Node Name:	testVirtual	Node ID:				
Node Type:	Virtual Node	Owner:	energilen 🗸			
Description:						
Parameters: Format: Variable I=Parameter1 Variable2=Parameter2Separate parameters with spaces.						

- Node Name: ワークフローノードの作成時に入力するノード名。 ノード名を編集するには、 ディレクトリツリーでノード名を右クリックし、ショートカットメニューから [Rename] を 選択します。
- ・ Node ID: タスクの送信時に生成される一意のノード ID で、編集できません。
- · Node Type: ワークフローノードの作成時に選択するノードタイプで、編集できません。
- Owner: ノードのオーナーです。デフォルトでは、現在ログインしているユーザーが新規作成したノードのオーナーになります。オーナーを変更するには、入力ボックスをクリックしてオーナーの名前を入力するか、または直接別のユーザーを選択します。

注:

別のユーザーを選択する場合、そのユーザーは現在のプロジェクトのメンバーである必要が あります。

- · Description: 通常、ビジネスとノードの目的を説明するために使用します。
- Parameter: タスクのスケジューリング時にコード内の変数に値を割り当てるために使用します。

たとえば、変数 "pt=\${datetime}" を使用してコード内で時間を指定する場合、この変数に 値を割り当てることができます。 割り当てる値には、スケジューリングのビルトイン時間パラ メーター "datetime=\$bizdate" を使用できます。

さまざななノードタイプのパラメーター値の割り当て形式

- ODPS SQL, ODPS PL, ODPS MR タイプ: Variable name 1=Parameter 1 Variable
 name 2=Parameter 2... 複数のパラメーターはスペースで区切ります。
- SHELL タイプ: Parameter 1 Parameter 2... 複数のパラメーターはスペースで区切ります。

頻繁に使用される時間パラメーターは、ビルトインスケジューリングパラメーターとして提供さ れます。パラメーターの詳細については、「パラメーターの設定」をご参照ください。

1.5.2 パラメーターの設定

スケジュールされた時間にタスクが自動的に実行される際、タスクが環境変化に動的に対応でき るように、DataWorks にはパラメーターの設定機能が搭載されています。 パラメーターの設定 を行う前に、次の2つの問題点について、特に注意してください。

・イコール "=" の両端にスペースを挿入することはできません。 正しい例: bizdate=\$bizdate

Basics ⑦				
Node Name:	testVirtual	Node ID:		
Node Type:	Virtual Node	Owner:	wingdam v	
Description:		no space is added on both sides of	the equal sign	
Parameters:	bizdate=\$bizdate		C)

複数のパラメーターがある場合は、スペースで区切ります。

Basics ⑦								
Node Name:	testVirtual	Node ID:						
Node Type:	Virtual Node	Owner:	wangalan					
Description:	if there are multiple parameters, each parameter is separated by spaces.							
Parameters:	bizdate=\$bizdate datetime=\${yyyymmdd}			0				

システムパラメーター

DataWorks には2つのシステムパラメーターがあり、次のとおり定義します。

- ・ \${bdp.system.cyctime}: インスタンスのスケジュールされた実行時間を定義します。デフォルトの形式は、yyyymmddhh24missです。
- ・ \${bdp.system.bizdate}: インスタンスの計算が行われる営業日を定義します。 デフォルトの営業日は、実行日の一日前です。デフォルトの表示形式は、yyyymmdd です。

定義に基づいたランタイムの計算式と営業日の数式は次のとおりです。Runtime = Business date + 1

システムパラメーターを使用するには、編集ボックスでシステムパラメーターを設定せずに、 コードで直接 **\${bizdate}** を参照します。システムでは、コードでシステムパラメーターの参照 フィールドが自動的に置換されます。



定期的なタスクのスケジューリング属性は、スケジュールされたランタイムを使って設定されま す。したがって、インスタンスのスケジュールされたランタイムに基づいて営業日をバックト ラックし、インスタンスのシステムパラメーター値を取得することができます。

例

毎日 00:00 から 23:59 まで毎時間、ODPS_SQL タスクが実行されるように設定します。 コード でシステムパラメーターを使うには、次の文を実行します。

```
insert overwrite table tb1 partition(ds ='20180606') select
c1,c2,c3
from (
select * from tb2
where ds ='${bizdate}');
```

非 Shell ノードのスケジューリングパラメーターの設定

注注:

SQL コードの変数名には、a-z、A-Z、数字、およびアンダースコアを使用することができま す。変数名が "date" の場合、値 "\$bizdate" は自動的にこの変数へ割り当てられます。スケ ジューリングパラメーターの設定で値を割り当てる必要はありません。 もし別の値が割り当て られても、コードではデフォルトで自動的に値 "\$bizdate" が割り当てられるため、この値は コードでは使用されません。

非 Shell ノードでは、\${variable name} (関数が参照されることを示します) をコードに追加し ます。その後、スケジューリングパラメーターへ割り当てる特定の値を入力します。

たとえば、ODPS SQL ノードの場合、**\$**{variable name} をコードへ追加し、ノードのパラメー ター項目 "variable name =built-in scheduling parameter" を設定します。

コードで参照されているパラメーターでは、スケジューリング中に解決済みの値を追加する必要があります。



2. 値は、コードで参照されている変数へ割り当てる必要があります。 値の割り当て規則

```
は、variable name = parameter です。
```

Basics ⑦				
Node Name:	insert_data	Node ID:		
Node Type:	ODPS SQL	Owner:	wangdan 🗸	
Description:	bizdate=\$bizdate			
Parameters:	Format: Variable1=Parameter1 Variable2=Parameter	r2Separate parameters with		0

Shell ノードのスケジューリングパラメーターの設定

Shell ノードのパラメーター設定手順は、非 Shell ノードの設定手順と似ていますが、規則は異なります。
Shell ノードでは、変数名をカスタマイズすることはできません。変数名は、"\$1、\$
2、\$3..."にする必要があります。

たとえば、Shell ノードの場合、コード内の Shell 構文宣言は、\$1 です。スケジューリングの ノードパラメーターの設定は、\$xxx (ビルトインスケジューリングパラメーター) です。 つま り、コード内の \$1 を置換するため、\$xxx の値が使用されます。

コードで参照されているパラメーターでは、スケジューリング中に解決済みの値を追加する必要があります。



∐ 注:

Shell コードでは、パラメーターの数が 10 になる場合、\${10} を使って変数を宣言する必要 があります。

 値は、コードで参照されている変数へ割り当てる必要があります。 値の割り当て規則は、パ ラメーター1、パラメーター2、パラメーター3...(置換された変数は、パラメーターの位置 に基づいて解決されます。たとえば、\$1はパラメーター1に解決されます。

0

変数値が固定値の場合

SQL ノードを例にします。 コードの **\${variable name}** では、ノードのパラメーター項目 **"variable name = "fixed value"** を設定します。

 $\exists - F$: select xxxxx type=' \${type}'

スケジューリング変数に割り当てられる値: type="aaa"

スケジューリング中、コードの変数は、type="aaa" に置換されます。

変数値がビルトインスケジューリングパラメーターの場合

SQL ノードを例にします。 コード内の **\${variable name}** では、ノードのパラメーター項目 " **variable name=scheduling parameter**"" を設定します。

 $\exists - F$: select xxxxx dt=\${datetime}

スケジューリング変数に割り当てられる値: datetime=\$bizdate

スケジューリング中、今日が2017 年 7 月 22 日の場合、コードの変数は dt=20170721 に置換 されます。

ビルトインスケジューリングパラメーターの一覧

\$bizdate: 形式 yyyymmdd で示す営業日 注記: このパラメーターは広く使用され、ルーチンス ケジューリングではデフォルトでは前日に日付となります。

たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$bizdate です。 今日が 2017 年 7 月 22 日とします。 ノードが今日実行され る場合、\$bizdate は pt=20170721 に置換されます。

たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$gmtdate です。 今日が 2017 年 7 月 22 日とします。 ノードが今日実行され る場合、\$gmtdate は pt=20170722 に置換されます。 **\$cyctime:** タスクのスケジュール時間 毎日のタスクに対し、時間がスケジュールされていない場合、cyctime は現在の日の 00:00 になります。時間の時、分、秒は正確で、これらは通常、時間単位または分単位のスケジューリングタスクに使用されます。例: cyctime=\$cyctime

🧾 注:

\$[]と\${}を使って設定された時間パラメーターの違いに注意してください。\$bizdate: 営業 日です。デフォルトでは、現在より1日前の日です。\$cyctime: タスクのスケジュールされた 時間です。毎日のタスクの対し、時間がスケジュールされていない場合、タスクは現在の日の 00:00 に実行されます。時間の時、分、秒は正確で、これらは通常、時間単位または分単位の スケジューリングタスクに使用されます。たとえば、タスクが現在の日の 00:30 に実行される ようにスケジュールされている場合、スケジュール時間は、yyyy-mm-dd 00:30:00 です。[] を使って時間パラメーターを設定した場合、cyctime は実行のベンチマークとして使用されま す。この使用の詳細については、次の手順をご参照ください。時間の計算方法は、Oracle と 同じです。データ作成中、パラメーターは選択した営業日プラス1日に置換されます。たとえ ば、営業日 20140510 がデータ作成中に選択された場合、cyctime は 20140511 に置換されま す。

\$jobid: タスクが属しているワークフローの ID 例: jobid=\$jobid

\$nodeid: ノードの ID 例: nodeid=\$nodeid

\$taskid: タスクの ID、つまりノードインスタンスの ID です。 例: taskid=\$taskid

\$bizmonth:形式 yyyymm で示す営業月

- ・ 営業日の月が現在の月と同じ場合、**\$bizmonth** = 営業日の月マイナス1です。同じでない場合は、**\$bizmonth** = 営業日の月です。
- たとえば、ODPS SQL ノードのコードでは、pt=\${datetime}です。 ノードのパラメーター
 設定は、datetime=\$bizmonthです。 今日が 2017 年 7 月 22 日とします。 ノードが今日
 実行される場合、\$bizmonth は pt=201706 に置換されます。

\$gmtdate: 形式 **yyyymmdd** で示される現在の日付です。 このパラメーターの値は、デフォル トでは現在の日付です。 データ作成中、入力である **gmtdate** は営業日プラス**1**です。

カスタムパラメーター \${…}パラメーターの説明:

- ・ \$bizdate に基づいてカスタムした時間形式では、yyyy は 4 桁の年、yy は 2 桁の月、mm は月、dd は日を示します。パラメーターを組み合わせることができます。たとえば、\${yyyy }、\${yyymm}、\${yyyymmdd} および \${yyyy-mm-dd} です。
- ・ **\$bizdate** は年、月、日です。 カスタムパラメーター **\$**{……} でも、年、月、日のみを表現します。

・ 特定の期間の前後の範囲を取得する方法

次の N 年: \${yyyy+N}

前の N 年: \${yyyy-N}

次の N 月: \${yyyymm+N}

前のN月: \${yyyymm-N}

次の N 週間: \${yyyymmdd+7*N}

前のN週間: \${yyyymmdd-7*N}

次のN日:\${yyyymmdd+N}

前のN日: \${yyyymmdd-N}

\${yyyymmdd}:形式 yyyymmdd で示す営業日です。 値は、**\$bizdate**の値と一致します。

- このパラメーターは広く使用され、ルーチンスケジューリングではデフォルトで、前日の日付です。このパラメーターの形式をカスタムすることができます。たとえば、\${yyyy-mm-dd}の形式を yyyy-mm-dd にカスタムできます。
- たとえば、ODPS SQL ノードのコードでは、pt=\${datetime}です。 ノードのパラメーター 設定では、datetime=\${yyyymmdd}です。 今日が 2013 年 7 月 22 日とします。 ノードが 今日実行される場合、\${yyyymmdd}は pt=20130721 に置換されます。

\${yyyymmdd-/+N}: yyyymmdd プラス、またはマイナス N 日

\${yyyymm-/+N}: yyyymm プラス、またはマイナス N 月

\${yyyy-/+N}: 年 (yyyy) プラス、またはマイナス N 年

\${yy-/+N}:年(yy) プラス、またはマイナス N 年

yyyymmdd は営業日を示し、yyyy-mm-dd のように区切ることができます。 上記のパラメー ターでは、営業日の年、月、日から取得されます。

例:

- ODPS SQL ノードのコードでは、pt=\${datetime}です。ノードのパラメーター設定では、 datetime=\${yyyy-mm-dd}です。 今日が 2018 年 7 月 22 日とします。ノードが今日実行 される場合、\${yyyy-mm-dd}は pt=2018-07-21 に置換されます。
- ODPS SQL ノードのコードでは、pt=\${datetime}です。ノードのパラメーター設定では、 datetime=\${yyyymmdd-2}です。 今日が 2018 年 7 月 22 日とします。ノードが今日実行 される場合、\${yyyymmdd-2} は pt=20180719 に置換されます。

- ODPS SQL ノードのコードでは、pt=\${datetime}です。ノードのパラメーター設定では、 datetime=\${yyyymm-2}です。 今日が 2018 年 7 月 22 日とします。ノードが今日実行される場合は、\${yyyymm-2} は pt=201805 に置換されます。
- ODPS SQL ノードのコードでは、pt=\${datetime}です。ノードのパラメーター設定では、 datetime=\${yyyy-2}です。今日が 2018 年 7 月 22 日とします。ノードが今日実行される 場合は、\${yyyy-2} は pt=2018 に置換されます。

ODPS SQL ノード設定では、複数のパラメーターに値が割り当てられます。たとえば、 startdatetime=\$bizdate enddatetime=\${yyyymmdd+1} starttime=\${yyyy-mm-dd} endtime=\${yyyy-mm-dd+1} です。

```
例: ($cyctime=20140515103000 とします)
```

- \$[yyyy] = 2014, \$[yy] = 14, \$[mm] = 05, \$[dd] = 15, \$[yyyy-mm-dd] = 2014-05-15, \$[hh24:mi:ss] = 10:30:00, \$[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00
- \$[hh24:mi:ss 1/24] = 09:30:00
- \$[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
- \$[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-1509:30:00
- \$[add_months(yyyymmdd,-1)] = 2014-04-15
- \$[add_months(yyyymmdd,-12*1)] = 2013-05-15
- \$[hh24] =10
- \$[mi] =30
- パラメーター **\$cyctime** のテスト方法

インスタンスを実行した後、ノードを右クリックし、[check the node attribute] を行いま す。 スケジュール時間が、インスタンスを定期的に実行する時間になっているかどうかを確認し ます。

パラメーター値の後の結果は、スケジュール時間マイナス1時間に設定されます。

よくある質問

 Q:テーブルパーティションの形式が pt=yyyy-mm-dd hh24:mi:ss ですが、スケジューリン グパラメーターではスペースが使えません。 \$[yyyy-mm-dd hh24:mi:ss] の形式で設定し た方が良いですか。

A: カスタム変数パラメーター datetime=\$[yyyy-mm-dd] と hour=\$[hh24:mi:ss] を使ってそれぞれの日と時間を取得します。その後、コードで結合して pt="\${datetime} \${hour} }" を作成します。(2つのカスタムパラメーターはスペースで区切ります。)

 ・ Q: コードのテーブルパーティションが pt="\${datetime} \${hour}" です。 実行中の最終時間 のデータを取得するには、カスタム変数パラメーター datetime=\$[yyyymmdd] と hour= \$[hh24-1/24] を使ってそれぞれの日と時間を取得することができます。 ただし、インスタン スの実行が 00:00 の場合、計算結果は前日の 23:00 ではなく現在の日の 23:00 になってしま います。 この場合、どのような処置をとることができますか。

A: datatime の形式を **\$[yyyymmdd-1/24]** に変更し、時間の形式を **\$[hh24-1/24]** のまま にします。計算結果は次のとおりとなります。

- インスタンスのスケジュール時間が 2015-10-27 00:00:00 の場合、\$[yyyymmdd-1/
 24] の値と \$[hh24-1/24] の値はそれぞれ、20151026 と 23 になります。これは、スケジュール時間マイナス1時間が、昨日に属する時間値だからです。
- インスタンスのスケジュール時間が 2015-10-27 01:00:00 の場合、\$[yyyymmdd-1/24] の値と \$[hh24-1/24] はそれぞれ、20151027 と 00 になります。これは、スケジュール時間マイナス 1 時間が、現在の日に属する時間値だからです。

DataWorks では4つの実行方法が提供されています。

- ・データ開発ページでの実行:正常に実行するためには、パラメーター設定ページで一時的な値の割り当てが必要になります。しかし、割り当て値はタスク属性としては保存されず、他3
 つの実行モードに影響を与えません。
- ・ 任意の周期での自動実行: パラメーター編集ボックスで設定する必要はありません。パラメー ターは、現行インスタンスでスケジュールされたランタイムへ自動的に置換されます。
- ・ テスト実行/データサプリメント実行:実行をトリガーする営業日を設定する必要があります。
 各インスタンスのシステムパラメーター値2つを取得するため、前述の数式によってスケジュールされるランタイムが求められます。

1.5.3 時間属性

時間属性設定ページは次の図で示すとおりです。
Schedule ⑦				
Schedule :	📀 Normal 🔵 Zero-łoad			
Error Rate this product :	0			
Validity Period :	1970-01-01	9999-01-01	8	
	validity Period of the task wil			
Pause Scheduling :				
Schedule Interval :	Dey			
Plan Time :	•			
Planned Time :	00:26			
	Note: The default planned tin	ne is randomly selected f	rom 0:00 to 0:30.	

ノード状態

- Normal:次のスケジューリングサイクルに基づいてノードがスケジュールされています。このオプションはデフォルトで選択されています。
- Zero-load: このオプションを選択すると、次のスケジューリングサイクルに基づいてノード が設定されスケジュールされます。しかし、タスクがスケジュールされると、タスクを実行せ ずに成功結果が直接返されます。
- Error retries: ノードにエラーが発生した場合、ノードを再実行することができます。デフォルトのエラーでは自動的に2分間隔で3回再実行されます。
- Suspend scheduling: このチェックボックスを選択すると、次のスケジューリングサイクル に基づいてノードが設定されスケジュールされます。しかし、このタスクがスケジュールされ ると、タスクを実行せずに失敗結果が直接返されます。タスクが中断されているが後で実行 する場合に使用します。

スケジューリング間隔

DataWorks では、タスクが正常に送信されると、下にあるスケジューリングシステムはタスク の時間属性に基づいて、次の日から毎日インスタンスを生成します。実行結果と依存関係にある アップストリームインスタンスのタイムポイントに基づいてインスタンスを実行します。23:30 以降に正常に送信されたタスクの場合、インスタンスは3日目から生成されます。

注注:

タスクを毎週月曜日に実行する必要がある場合、ランタイムが月曜日に設定されている場合の みタスクが実行されます。 ランタイムが月曜日に設定されていない場合、タスク (直接成功と設 定されている)が偽りで実行されます。この理由から、テスト実行またはデータサプリメント実行で週単位のスケジュールタスクでは営業日=ランタイム-1を設定します。

サイクル通りに実行されるタスクでは、依存関係の優先順位は、時間属性の優先順位よりも高 くなります。したがって、時間属性で指定した時間になると、タスクインスタンスはすぐに実 行されず、まずすべてのアップストリームインスタンスが正常に実行されたかどうかを確認しま す。

- ・依存関係にある上位インスタンスすべてが正常に実行されておらず、スケジュールされたランタイムになっている場合、インスタンスは実行以外の状態を維持します。
- ・依存関係にある上位インスタンスすべてが正常に実行されておらず、スケジュールされたランタイムになっている場合、インスタンスは実行以外の状態を維持します。
- ・依存関係にある上位インスタンスすべてが正常に実行されており、スケジュールされたランタイムになっている場合、インスタンスはリソース待ち状態となり、実行準備完了状態となります。

日単位のスケジューリング

日単位でスケジュールされたタスクは、毎日1回自動的に実行されます。 周期タスクを作成する と、デフォルトでタスクは毎日 00:00 に実行されるように設定されます。 必要に応じて、別のラ ンタイムを指定することができます。 たとえば、次の図に示すとおり、ランタイムを毎日 13:00 に指定することができます。

- Regular Scheduling の選択が解除されている場合、毎日のタスクインスタンスのスケジュール時間は、YYYY-MM-DD の現在の日付となります。0:00 から 0:30 の間でランダムに生成されるデフォルトのスケジューリング時間となります。
- Regular Scheduling が選択されている場合、毎日のタスクインスタンスのスケジュール時間は、YYYY-MM-DDの現在の日付となり、スケジュール時間は HH:MM:SS となります。 スケジュールされたタスクは、上位タスクが正常に実行され、スケジュールされた時間になった時のみ実行されます。この条件のどちらかが満たされていない場合、タスクを実行することはできません。条件に順序はありません。

Validity Period :	1970-01-01	9999-01-01	
		ive date effect and automa	
	validity Period of the task will not		
Pause Scheduling :			
Schedule Interval :	Day		*
Plan Time :			
Planned Time :	13:00		
CRON Expression :	00 00 13 * * ?		
Depend on Last Interval :	0		

使用例:

インポート、統計処理、エクスポートタスクは、上記の図に示すとおり、すべて毎日ランタイム 13:00 に行われるタスクです。統計処理タスクはインポートタスクに依存し、エクスポートタス クは統計処理タスクに依存します。次の図では、これらの依存関係 (統計処理タスクの依存属性 設定、インポートタスクに設定されているアップストリームタスク)を示します。

上記の図にある設定に基づいて、スケジューリングシステムは、タスクのインスタンスを自動的 に生成し、次のとおり実行します。



週単位のスケジューリング

週単位でスケジュールされているタスクは、毎週特定の日の特定の時間に自動的に実行されま す。特定されていない日に到達した場合、システムはインスタンスを生成します。また、論理の 実行を行ったりリソースを消費したりせずに、直接インスタンスを正常に実行された状態と設定 し、ダウンストリームインスタンスが適切に実行されるようにします。

Schedule ⑦	
Schedu	le: 💿 Normal 🔘 Zero-Ioad
Error Rate this produ	et: 🗌 🕐
Validity Peri	ud : 1970-01-01 - 9999-01-01 📾
	Note: The schedule will be effective date effect and automatic scheduling, on the other hand, validity
	Period of the task will not be automatic scheduling, manual scheduling.
Pause Scheduli	ng:
Schedule Interv	al: Week 🗸 🗸
Plan Tir	ne: 🗸
Specified Tin	ne: Monday × Friday × ×
Planned Tir	ne : 13:00 ③
CRON Expressi	n: 00 00 13 ? * 1,5
Depend on Last Interv	al:

上記の図に示すとおり、毎週月曜日と金曜日に生成されたインスタンスはスケジュールどおり実 行され、毎週火曜日、水曜日、木曜日、土曜日、日曜日に生成されたその他のインスタンスは、 正常に実行されたと設定されます。

上記の図にある設定に基づいて、スケジューリングシステムは、タスクのインスタンスを自動的 に生成し、次のとおり実行します。



月単位のスケジューリング

月単位でスケジュールされているタスクは、毎月特定の日の特定の時間に自動的に実行されま す。特定されていない日に到達した場合、システムはインスタンスを毎日作成します。また、論 理の実行を行ったりリソースを消費したりせずに、直接インスタンスを正常に実行された状態と 設定し、下位インスタンスが適切に実行されるようにします。

Schedule ⑦		
Schedule :	: 💿 Normal 🔿 Zero-load	
Error Rate this product :		
Validity Period :	: 1970-01-01 🛱	
	Note: The schedule will be effective date effect and automatic scheduling, on the other hand, validity	
Pause Scheduling :		
Schedule Interval :	: Month v	
Plan Time :		
Specified Time :	: Day 1 × ·	
Planned Time :	: 00:00 ③	
CRON Expression :	: 00 00 00 15*?	
Depend on Last Interval :		

上記の図に示すとおり、毎月1日に生成されたインスタンスはスケジュールどおりに実行され、 その月の残りの日に生成されたインスタンスは、正常に実行されたと設定されます。

上記の図にある設定に基づいて、スケジューリングシステムは、タスクのインスタンスを自動的 に生成し、次のとおり実行します。

Scheduling task definition		Scheduling task instance	
	 business date : 2016-12-31	business date : 2017-01-01 至2017-01-30	bușiness date :2017-01-31
Weekly scheduling 00 00 00 1 * ?	2017-01-01 00:00:00	2017-01-02至31 00:00:00 (Running case) 土田 う	2017-02-01 00:00:00

時間単位のスケジューリング

時間単位でスケジュールされたタスクは、毎日 Nx1 時間ごとに実行されます。たとえば、毎日1:00 から 4:00 までの間、1 時間ごとに実行されます。

_____注:

実行間隔は、左クローズと右クローズ原理に基づいて計算されます。 たとえば、時間単位でス ケジュールされたタスクは、0:00 から 3:00 までの間、1 時間ごとに実行されるように設定さ れている場合、時間の間隔は [0:00, 3:00] で、間隔は 1 時間です。 スケジューリングシステム は、毎日インスタンスを 4 つ生成し、0:00、1:00、2:00、3:00 に実行します。

Error Rate this product :				
Validity Period :	1970-01-01 - 9	9999-01-01		
			atic scheduling, on the other hand, validity Period of t	
	task will not be automatic schedulir			
Pause Scheduling :				
Schedule Interval :	Hour			
Plan Time :				
• Start Time : 00:00	⊙ Interval : 1 ∨ h	End Time : 23:59	9 (3	
O Specified Time :	0:00 × Y			
CRON Expression :	00 00 00-23/1 * * ?			
Depend on Last Interval :				

上記の図に示すとおり、自動スケジューリングは毎日 0:00 から 23:59 の間の 6 時間ごとにトリ ガーされます。 したがって、スケジューリングシステムは、次のとおり、タスクのインスタンス を自動的に生成し、実行します。

scheduling task definition	Schee	Juling task nce		
	business date: 20	17-01-10 There 4	examples.	
小时任务 00 00 00-23/6 * * ?	小时任务 2017-01-11 00:00	小时任务 2017-01-11 06:00	小时任务 2017-01-11 12:00	小时任务 2017-01-11 18:00
			云河社区	yq.aliyun.cor

分単位のスケジューリング

分単位でスケジュールされたタスクは、次の図に示すとおり、毎日 N x 1 分 ごとに実行されま す。

タスクは 0:00 から 23:00 までの間、30 分ごとにスケジュールされています。

Schedule ?		
Schedule :	💿 Normal 🔵 Zero-Ioad	
Error Rate this product :	0	
Validity Period :	1970-01-01 - 99	99-01-01 📾
	Period of the task will not be automat	
Pause Scheduling :		
Schedule Interval :	Minute	
Plan Time :		
Start Time :	00:00	
Interval :	30 ③	min
End Time :	23:00 ③	
CRON Expression :	00 */30 00-23 * * ?	

現在、分単位のスケジューリングは、最低**5**分単位での設定をサポートしています。時間の表現 は選択式で、編集することはできません。

Schedule ⑦		
Schedule :	📀 Normal 🔵 Zero-load	
Error Rate this product :	0	
Validity Period :	1970-01-01	9999-01-01
	30	
Pause Scheduling :	0	
Schedule Interval :	5	
Plan Time :	15	
Start Time :	20	
Interval :	30	min
End Time :	23:59	
CRON Expression :	00 */30 00-23 * * ?	

よくある質問

Q: 上位タスク A が時間単位でスケジュールされ、下位タスク B が日単位でスケジュールされて います。さらに毎日タスク A が完了した後にタスク B を実行する必要がある場合、タスク A と タスク B はお互いに依存関係にありますか。

A: 日単位のタスクは、時間単位のタスクに依存します。 タスク A が時間単位でスケジュールさ れている場合、日単位でスケジュールされているタスク B は不定期にスケジュールされたものと なります。タスク A とタスク B はお互いに依存関係にあります。タスク B は毎日 24 時間タスク A が正常にインスタンスを実行した後に実行されます。 (依存関係の設定に関する詳細は、スケ ジューリング依存関係の説明をご参照ください。) したがって、各周期タスクはそれぞれに依存 しており、各タスクのスケジューリングサイクルは、タスクの時間属性によって判断されます。

Q: タスク A を毎時間実行し、タスク B は 1 日 1 回実行し、さらにタスク A が 1 回目に正常に実行された後にタスク B が開始する場合、 どのように設定すれば良いですか。

A: タスク A を設定する際、[Previous Cycle Dependent] と [Current Node] を選択し、タス ク B のスケジュール時間を 0:00 に設定します。 このように、自動的に毎日スケジュールされる インスタンスでタスク B のインスタンスは、タスク A の 0:00 インスタンスのみ、つまりタスク A の 1 つ目のインスタンスにのみ依存することとなります。

Q: タスク A を毎週月曜日に実行し、タスク B はタスク A に依存させる場合、タスク B が毎週月 曜日に実行されるよう、どのように設定すれば良いですか。

A: タスク B の時間属性をタスク A の時間属性と同じように設定します。スケジューリングサイ クルを [Weekly] と [Monday] に設定します。

Q: タスクのインスタンスは、タスクを削除した場合、影響をうけますか。

A: ある期間実行した後にタスクを削除した場合、そのインスタンスは残ります。これはスケ ジューリングシステムが時間属性に基づいて、1 つ以上のタスクのインスタンスをまだ生成して いるからです。このため、タスクが削除された後にインスタンスがトリガーされた場合、必要と されるコードが見つからないため、次のエラーメッセージが表示されます。

	[TASK=100021143283] execute task failed, nulljava.lang.NullPointerException
⊗ 7.17 20:22:00~20:22:00 (dur 0s)	
gateway:null	

Q:毎月末日に毎月のデータを計算する場合、どのようにすれば良いですか。

A: 現在、システムでは毎月末日をランタイムとして設定することはできません。 したがって、 タスクを毎月 31 日に実行するように設定する場合、スケジューリングは 31 日間ある月には 1 日 トリガーされ、インスタンスが生成されます。他の日のインスタンスは正常に実行されたと設定 されます。

月次統計では、毎月1日に前月のデータを計算することを推奨します。

1.5.4 ノードコンテキスト

ノードコンテキスト (Node Context) は、上位ノードと下位ノード間でパラメーターを転送する ために使用します。ノードコンテキスト機能では、最初に上位ノードの出力パラメーターと値を 定義し、次に下位ノードの入力パラメーターを定義するのが基本的な使い方です (入力パラメー タ値には、上位ノードの出力パラメーターが参照されます)。下位ノードでこのパラメーターを 使用することで、上位ノードから転送される値を取得できます。

ノードコンテキストのパラメーターは、次の図に示すとおり、特定ノードの[Schedule] > [Node Context] で設定します。

	nter an output name		+					
Output Na	me	Output Table Nam e	Downstream Node Nam e	Downstream Node I D	Owne r	Source	Actions	
bigdata_do	oc.test ©	- @				Added Manually		
bigdata_Di t	OC.30135300_ou	- C				Added by Defaul t		
Node Cont	text ⑦	Add						
NO.	Parameter Name	Value Of The Source	None	Parent Node ID	Sourc	e Actions		
The Node Out	tput Parameters	Add						
No.	Parameter Na	me -	Type Value	Description	Sou	rce Actio	ons	
			None					

出力パラメーター

[The Node Output Parameters] は[Node Context] で定義します。 出力パラメーター値に は、「Constant」と「Variable」の2種類あります。 「Constant」は固定文字列です。 「Variable」はシステムでサポートされているグローバル変数です。 上位ノードで出力パラ メーターが送信された後、その出力パラメーターを入力パラメーター値として下位ノードで再利 用できます。

🗎 注:

現行ノード (PyODPS ノードなど) に定義されている出力パラメーターに対して、内部コードを 記述して値を割り当てることはできません。

Node Co	ntext ⑦					
The Node Ir	nput Parameters	dd				
No.	Parameter Name	Value Of The Source	Description	Parent Node ID	Source	Actions
			None			
The Node O	utput Parameters	Add				
N o.	Parameter Name	Туре	Value	Description	Source	Actions
1	output_const	Constant V	abc	example of constant vi	Added M y	Manuall Save Cancel

パラメーターは次のように記述します。

フィールド	説明	注意
No.	[No.] の値はシステムに よって生成され、自動 的に加算されます。	N/A
Parameter name	出力パラメーター名を 定義します。	N/A
Туре	パラメータータイプで す。	出力パラメーター値には、Constant と Variable の 2 種類あります。
Value	ソースの値です。	 [Type] でConstant が選択されている 場合、文字列を直接入力することができ ます。 [Type] で Variableが選択されている 場合、システム変数、スケジュールビル トインパラメーター、カスタマイズパラ メーター \$ {} と \$ […] を使用できま す。
Description	パラメーターの簡単な 説明です。	N/A

Action	「Edit」および	「Edit」および「Delete」は、下位ノー
	「[Delete」 を選択で	ドと依存関係がある場合は使用できませ
	きます。	ん。上位ノードへの参照を追加する前に、
		上位ノードの出力パラメーターが正しく定
		義されていることを確認してください。

入力パラメーター

[The Node Input Parameters] は、依存関係のある上位ノードの出力値の参照を定義するため に使用します。他のパラメーター同様に、ノード内で使用することができます。

- ・ [The Node Input Parameters] の定義
 - 1. [Dependencies]で、依存関係のある上位ノードを追加します。

Dependencies ⑦							
Upstream Node Enter an output name or output ta	ible name 👻 🕂 Use The Workspace Root Nod	e Automatic recommended					
Upstream Node Output Name	Upstream Node Output Table Name	Node Name	Upstream Node ID	Owner	Source	Actions	
				100000	Added Manually		

2. [Node Context] > [The Node Input Parameters を選択し、上位ノードの値を参照する 入力パラメーターを定義します。

Node Contex	Node Context (0)													
The Node Input Parameters Add														
No.	Parameter Name	Value Of The Source	Description	Parent Node ID	Source	Actions								
	output_const	Please select ~			Added Manually	Save Cancel								

パラメーターは次のように記述します。

フィールド	説明	注意
No.	[No.] の値はシステムによって生 成され、自動的に加算されます。	N/A
Parameter name	入力パラメーター名を定義しま す。	N/A
Value Of the Source	パラメーターの値ソースで、上位 ノードの値を参照します。	上位ノードが実行中の特定のパラ メーター値です。
Description	パラメーターの簡単な説明です。	上位ノードから自動的に解析され ます。
Parent Node ID.	親ノード ID です。	上位ノードから自動的に解析され ます。
Action	「 Edit 」 および 「 Delete 」 を選 択できます。	N/A

• 入力パラメーターの使用

定義された入力パラメーターの再利用する際の形式は、他のシステムと同様です。形式 は、\${input parameter name} です。たとえば、Shell ノードの参照は次の図に示すとお りです。



サポート対象のグローバル変数

・ システム変数

\$ {projectid}: Project ID \$ {project name}: MaxCompute project name \$ {nodeid}: Node ID \$ {gmtdate}: 00:00:00 at the instance date,format: 'yyyy-mm-dd 00:00 :00'. \$ {taskid}: Instance Task ID \$ {seq}: Task instance sequence number,represents the instance's sequence number in the same node on current day. \$ {cyctime}: instance time \$ {status}: Status of instance-Success, Failure \$ {bizdate}: Business Date \$ {finishtime}: Instance End Time \$ {taskType}: Instance Status—NORMAL,MANUAL,PAUSE,SKIP,UNCHOOSE, SKIP_CYCLE \$ {nodeName}: Node name

パラメーター設定の詳細は、「パラメーターの設定」をご参照ください。

例

ノード "test22" は、ノード "test223" の上位ノードです。 ノード "test22" で [Node Context] > [The Node Output Parameters] を選択し、設定を行います。 たとえば次の図に 示すとおり、パラメーター名をdate1、 値を \${yyyymmdd} にして、 [Run] をクリックします。

<u>ا</u> م		ि हि 🔍	A 🖬 🗱												
1	{														
	"con	×													
		Dependencie	s @												
		Upstream Node	Enter an output name	e or output table name 👻 🔤	+ Use The	Workspace Root N	ode Automatic recomm	Automatic recommended							
		Upstream No	de Output Name	Upstream Node Output Ta	ble Name	Node Name	Upstream Node ID	Owner	Source	Actions					
8 9								100001	Added Manually						
10 11 12		Output Enter	ran output name												
13 14		Output Name		Output Table Name	Downstream Nod	ie Name	Downstream Node ID	Owner	Source	Actions					
15 16									Added by Default						
17 18 19		Node Contex	kt @												
20 21 22		The Node Input F	Parameters Add												
23 24		No.	Parameter Name	Value Of The Source	De	scription	Parent Node ID	Source	Actions						
25 26 27 28															
29 30		The Node Output	t Parameters Add												
	}, "tvp	No. Pr	arameter Name	Туре	Value		Description	Source	e Action	IS					
33 34	"ver "ord	1	date1	Variable	∽ \${yyyymn	mdd}	date	Added	Manually Save	Cancel					

ノード "test22" を正常に送信した後、下位ノード "test223" を設定します。

道注:

"test223" の [Dependencies] > [Upstream Node Output Name] が "test22" の[Dependencies] > [Output Name] と同じであることを確認します。

[Node Context] > [The Node Input Parameter] > [Parameter Name] の順に選択 し、"test22" のパラメーター名 "date1" を 入力します。[Value Of The Source] のドロップ ダウンにオプションが表示されます。 該当するソースを選択し、[Save] を選択します。

Upstream Node Output Name	Upstream Node Output Tab	le Name	Node Name	Upstream Node ID	Owner	Source	Actions
			Test22	700001940205	alidocs	Added Manually	
Output Enter an output name							
Output Name	Output Table Name	Downstream No	de Name Downstream Node ID		Owner	Source	Actions
	- Ø					Added by Default	
Node Context ⑦ The Node Input Parameters							
No. Parameter Name	Value Of The Source	Descripti	on	Parent Node ID	Source	Actions	
1 date1	Please select	 			Added Manually	Save Cancel	

1.6 設定管理

1.6.1 設定管理の概要

設定管理は、DataStudio インターフェイスの設定で、コード、フォルダー、テーマ、モジュー ルの追加や削除などを行います。 データ開発の左下隅にある歯車をクリックして設定管理ページ へ移動します。



設定管理は、5つのモジュールに分かれています。 詳細は、以下のページをご参照ください。

- ・ 設定センター
- プロジェクト設定
- ・ テンプレート
- ・ テーマの管理
- ・ テーブルレベル

1.6.2 設定センター

設定センターでは、モジュール管理とエディタ管理を含む共通機能の設定を行います。

	Ξ
197	Configuration Center
ь	Project Configuration
10	Templetes
4	There Management
٠	Table Levels
	Beckup and Restore

モジュール管理

モジュール管理では、DataStudio インターフェイスの左列の機能モジュールヘモジュールを追 加または削除する操作を行います。クリックして、左側に表示する必要のある機能モジュールに フィルターをかけたり、ドラッグ&ドロップでモジュールの機能をソートしたりできます。

追加するモジュールにマウスを置くと、モジュールは青色に変わり、[Add] が表示されます。

Modules									
Added N	Nodules	Available Modules							
Data Development	Components	Add	Functions						
Queries	Runtime Log	Recycle Bin							
Manual Business Flows	Tables								

削除するモジュールにマウスを置くと、モジュールは赤色に変わり、[Remove] が表示されます。

Modules							
Added	Modules	Available Modules					
Data Development	Components	Public Tables	Functions				
Queries	Remove	Recycle Bin					
Manual Business Flows	Tables						

注:

テンプレート管理フィルターはすぐに、現在のプロジェクトに対して有効になります。すべての プロジェクトに対し有効にする場合は、[the above settings to apply to all projects] をク リックします。

エディタ管理

エディタではコードやキーワードの設定を行います。インターフェイスを更新することなく、リ アルタイムで設定が有効になります。

・ サムネイル表示

現在のインターフェイスコードはコードの右側に表示されます。図の陰がついているエリア は、現在表示されているエリアを示します。コードが長い場合は、マウスを上下に移動させる ことで表示されるコードエリアを切り替えます。

Sq cree	nte_table_ddl 🗙	sh testShell		Sq testSQL	٠	myComponent		testSQLComponent		ftp_sync		Business Flow		Solution		i se i
۳	🖽 🗗	۵ ٦	⊙												_	
1	CREATE TAB	LE IF NOT	EXIST	ods_user_	info_	d (Biol-		
2	2 uid STRING COMMENT '用户ID', String Comment 'L',															
3	3 gender STRING COMMENT '性别',															
-4	age_nange STRING COMMENT '年龄段', 概念															
5	zodiac S	TRING COMME	NT ' <u>#</u>	<u>e</u> .										1988		
6)													SUADO-		
7	PARTITIONE	D BY (8885		
8	dt STRIN															
9);															
10																
11																
12	CREATE TAB	LE IF NOT E	XISTS	ods_raw_l	.og_d	(
13	col STRI															
14)															

・エラーのチェック

現在のコードのエラー状態を確認します。 赤いエラーコードエリアにマウスを置くと、エ ラー特有のフィールド状態が表示されます。

Se crea	te_table_ddl 🔵	s testShe		ise testSQL	٠	≦` myComponent ×	f testSQLComponent ×	🖬 ftp_	syna 🛛 🔵	Busines	Flow	Solution			
•	5 F)	B	۲											08	м
54 55 56	CREATE TAB	LE IF NOT	EXISTS	rpt_user_i	1fo_d	(1999 1990			Schedul
57 58	region S	NG COMMENT TRING COMM	「用户」 ENT 「地	tD', U城,根据ip得	到',							101			
59 60 61 62 63	device S pv BIGIN gender S age_rang zodiac S	TRING COMM T COMMENT TRING COMM SE STRING C TRING COMM	ENT 'S 'pv', ENT 't OHMENT ENT '4	《编类型 ', [别], "年齡段', [座]								No.			Relationship
64 65) PARTITIONE														Ver
66 67	dt STRING);														
68 69	extraneou	s input 'C	TEATE'	expecting C	ops k	(eyword									
70 71	CTEATE TAB	1E IF NOT	EXIST	S test_data	iorks;	;									ucturi

・自動保存

現在編集中のコードを自動的にキャッシュして、編集中のページのクラッシュやコードが保 存されないことを防ぎます。 左側の **[Use server-saved code]**、または右側の **[Use locally cached code]** を選択します。

our edits were not saved last time and has been cached. Select a version	that you need.	
Code saved on the server by 王冄 at 2018-09-03 11:49	Code edited by wangdan at 2018-09-03 04 53 and cached locally	
1 CREATE TABLE IF NOT EXISTS ods_user_info_d (uid STRING COMMENT '用户ID', gender STRING COMMENT '性粉', age_range STRING COMMENT '生命' bit codiac STRING COMMENT '生命' codiac STRING COMMENT '生命' partitioned By (d d STRING); CREATE TABLE IF NOT EXISTS ods_raw_log_d (cod STRING) partitioned By (d d STRING	1 CREATE TABLE IF NOT EXISTS ods_user_info_d (2 uid STRING COMMENT '用户ID', 3 gender STRING COMMENT '性例', 4 age_range STRING COMMENT '性的', 5 zodiac STRING COMMENT '生能' 6) 7 PARTITIONED BY (8 dt STRING 9); 10 11 12 CREATE TABLE IF NOT EXISTS ods_raw_log_d (13 col STRING 14) 15 PARTITIONED BY (16 dt STRING 17)	

・ コードスタイル

コードスタイルは、お好みに応じて大文字、または小文字に設定することができます。 キー ワードを入力し、Enter を押して Lenovo ショートカットを使って必要なキーワードを入力 します。



コードのフォントサイズ

コードのフォントサイズは、最小 12 、最大 18 フォントをサポートします。ご自身のコード 書き込みの習慣や量に応じて、設定を変更します。

Sq inse	rt_data	×	👗 base_	cdp ×	89	create_table_ddl ●	testShell	Sq testSQL	•	☆ myComponent ×	testSQLComponent :	× DI	ftp_sync	•		
凹	Ē,	ſ	[٤]		Ð											
1 2 3 4 5	odp *** aut cre															Schedule
6 7	CREAT		BLE IF													Relationship

・ コードのヒント

コードのプロンプトはコードの入力の際に使用します。インテリジェントプロンプトの表示 は、次のセクションに分かれています。

- Space Smart Tip: Lenovo のキーワード、テーブル、フィールドを選択した後、スペース を追加します。
- keywords: プロンプトコードは入力したキーワードをサポートします。
- Syntax template: サポートされる構文テンプレートです。
- Project: Lenovo のプロジェクト名を入力します。
- Table: Lenovo で入力する必要のあるテーブルです。
- Field: このテーブルのフィールドに対するスマートプロンプトです。

・テーマ

テーマのスタイルは DataStudio インターフェイススタイルの設定です。現在は黒と白の両方 をサポートしています。

・ アプリケーション

上記のテンプレート管理設定とエディタ管理設定を現在の既存プロジェクトに適用します。

1.6.3 プロジェクト設定

プロジェクト設定には、パーティション日形式、パーティションフィールド名、一時テーブルプ レフィックス、アップロードテーブル (インポートテーブル) プレフィックスの 4 つの設定項目が あります。

	Ξ
👬 Confi	iguration Center
🖿 Proje	ect Configuration
Temp	plates
🛟 Then	ne Management
😂 Table	e Levels
🕞 Back	up and Restore

- Partition Date Format: デフォルトパラメーターです。コードのパラメーターの表示形式です。ご自身の要件に応じてパラメーターの形式を変更します。
- · Partition field naming: パーティションデフォルトフィールド名
- Temporary table prefix: "t_" で始まるフィールドは、デフォルトでは一時テーブルとして 認識されます。
- Upload (import table) prefix: DataStudio インターフェイスがテーブルをアップロードする際のテーブルの名前プレフィックスです。

1.6.4 テンプレート

テンプレート管理はノードが作成された後、デフォルトでコードの前に表示されるコンテンツで す。プロジェクト管理者は、必要に応じてテンプレートの表示スタイルを変更することができま す。

現在、ODPS SQL テンプレート、ODPS MR テンプレート、ODPS PL テンプレート、PERL テ ンプレートおよび SHELL テンプレートに対してタイトルが設定されます。

≡		
🙀 Configuration Center	Template	Actions
Project Configuration	ODPS SQL Template	Edit
Templates	ODPS MR Template	Edit
Theme Management	SHELL Template	Edit
📚 Table Levels		
Backup and Restore		

SQL ノードを例にとると、テンプレートの表示スタイルは次のとおりです。

	Data Developn 온 ඕ 다 C 운	Յ Թ	insert_dat	•	vi start	×®	create_table_ddl	x DI ftp_sync	•	🧐 insert_data	× 🗄
		Æ	8	(†	3	۲					
*	> Solution	88	10	dps so	1] *******	 					
民	✓ Business Flow	88									
÷.	🛩 🚣 base_cdp				time:2	8-31 1	15:59:06				
÷.	> 🔁 Deta Integration										

1.6.5 テーマの管理

テーブル管理にはたくさんのテーブルがあります。選択したトピックに応じて、テーブルは第二 レベルのサブフォルダに格納されます。 これらのフォルダはテーブルにまとめられ、これがテー マです。 管理者は、プロジェクトの要件に応じて複数のテーマを追加したり、テーブルの目的や 名前に応じて分類分けをしたり整理したりすることができます。

191	Configuration Center						
•	Project Configuration	Topic E		Parent Topic	Root Topic (Level 1 Topic by Default)	Add	
ī	Templates		Level 1 Repository Topic		Added B	Added At	
\$	Theme Management	+	one_level		3.9	2018-09-03 13:49:41	
۲	Table Levels						
8)	Backup and Restore						

1.6.6 テーブルレベル

テーブルレベルは、テーブルの物理的なレベルデザインです。 問題が発生しプロジェクトの影響 が正確に特定できず、それがオンライン操作の通常の操作へつながる場合、プロジェクトに対す るテーブルの重要性に応じて、問題が影響することを防ぐためにテーブルが分割されます。

the Configuration Center	Table Levels Table Level : Enter	Description : Enter Add	
Templates	Table Level	Description	Actions
Theme Management			
Table Levels			
Backup and Restore	Table Category Category Name : Enter	Description : Enter Add	
	Table Category	Description	Actions

プロジェクトにデフォルトの階層はありません。管理者はプロジェクトの目的や必要性に応じて 手動で追加する必要があります。

1.7 マニュアルビジネスフロー

1.7.1 マニュアルビジネスフローの紹介

マニュアルビジネスフロー (Manual Business Flow) では、作成されたすべてのノードを手動 でトリガーする必要があり、スケジューリングを使って実行することができません。 したがっ て、マニュアルビジネスフローでは、親ノードの依存関係とノードのローカルノード出力を設定 する必要がありません。

Deta Developa 🔒 🔃 📮 😋 🤬	👗 bese,cdp 🛛 🗙 🔄 fph	fgh 🛛 🛪 🔤 Runtime Log 🗙	📼 textShell 🗴 🔤 textSQL 🔹	🖆 myComponent 🛪 🏫 testSQLComponent :
Enter a file or creator name	□ ⊙ ⊙ ៧	3		
> Solution 🔡	v Data Internation	Development Road		
✓ Business Row 23				
🗘 🖬 base,cdp 🖉 🔂	Conta Sync			
👻 🚠 workshop	 Deta Development 			
 Data Integration The sync Mettill 09-02,1726 The sync determedia, demo203 The create, table_dd Mettill 09-03 Methods, and Method Methods (09-03) Methods, and Methods (09-03) Methods, and Methods (09-03) Methods, start Methods (09-03) Methods, start Methods (09-03) Methods, start Methods (09-03) 	COPS SQL Shell COPS MR Virual Node Py00PS SQL Component Node Node		• Vi start © • Er teatSHELL • Mr teatMR • Er teatM	Signi insertudota Signi insertudota write_result write_result With testVirtual SQLComponent

マニュアルビジネスフローインターフェイスの機能は次のとおりです。

No.	機能	説明
1	Submit	クリックして、現在のマニュアルビジネスフローにある すべてのノードを送信します。

No.	機能	説明
2	Run	クリックして、現在のマニュアルビジネスフローにある すべてのノードを実行しまうs。 手動タスクに依存関係 がない場合、これらのタスクは同時に実行されます。
3	Stop Run	クリックして、実行中のノードを停止します。
4	Publish	クリックしてタスク発行インターフェイスへ移動しま す。タスク発行インターフェイスでは、すでに送信され ているがまだ運用環境へ発行されていないノードの一部 またはすべてを発行することができます。
5	Go to O&M	クリックして O&M Center へ移動します。
6	Reload	クリックして、現在のマニュアルビジネスフローイン ターフェイスを再度読み込みます。
7	Auto Layout	クリックして、現在のマニュアルビジネスフローイン ターフェイスにあるノードを自動的にシーケンスしま す。
8	Zoom-in	クリックしてインターフェイスをズームインします。
9	Zoom-out	クリックしてインターフェイスをズームアウトします。
10	Query	クリックして、現在のマニュアルビジネスフローにある ノードを照会します。
11	Full Screen	クリックして、現在のマニュアルビジネスフローにある ノードを全画面表示モードで表示します。
12	Parameters	クリックして、パラメーターを設定します。フローパラ メーターの優先順位は、ノードパラメーターより高くな ります。パラメーターキーがパラメーターと一致する場 合、ビジネスフローパラメーターは優先的に設定されま す。
13	Operation Records	クリックして、現在のマニュアルビジネスフローにある すべてのノードの操作履歴を表示します。
14	Version	クリックして、現在のマニュアルビジネスフローにある すべてのノードの送信履歴と発行履歴を表示します。

1.7.2 リソース

リソース (Resource) は ODPS 特有のコンセプトです。 リソースは ODPS UDF または ODPS MR を使用する際に利用できます。

ODPS SQL UDF: UDF のコンパイル後、コンパイルされた jar パッケージを ODPS をアップ
 ロードする必要があります。この UDF を実行する際、ODPS は自動的に jar パッケージをダ

ウンロードし、ユーザーコードを抽出し、UDF を実行します。 jar パッケージのアップロー ドプロセスは、ODPS でリソースを作成するプロセスです。 jar パッケージは ODPS リソー スのタイプの 1 つです。

 ODPS MapReduce: MapReduce プログラムをコンパイル後、コンパイルされた jar パッ ケージをリソースとして ODPS ヘアップロードする必要があります。 MapReduce ジョブ を実行する際、MapReduce フレームワークはこの jar パッケージを自動的にダウンロード し、ユーザーコードを抽出します。

同様に、テキストファイル、ODPS テーブル、およびさまざまな圧縮パッケージ (.zip、.tgz 、.tar.gz、.tar および .jar など) をアップロードすることができます。 これで、UDF や MapReduce を実行する際リソースを読み込んだり使用したりできるようになります。

ODPSでは、リソースの読み込みや使用のために **API** が提供されています。 **ODPS** リソースの 次のタイプが利用可能です。

- ・ファイル
- アーカイブ:リソース名の拡張子で圧縮タイプを識別します。次の圧縮ファイルタイプがサポートされています。.zip、.tgz、.tar.gz、.tar および .jar です。
- ・ Jar: コンパイルされた Java jar パッケージです。

DataWorks では、リソースの作成プロセスはリソースの追加プロセスと同じです。 現 在、DataWorks では、3 つのタイプのリソースの視覚的な追加をサポートしていま す。jar、Python、ファイルリソースです。 新しく作成されたエントリは同じです。違いは次の とおりです。

- ・ Jar リソース: Java コードをオフライン Java 環境でコンパイルし、コードを jar パッケージ へ圧縮し、ODPS へ jar リソースとしてパッケージをアップロードする必要があります。
- ・スモールファイル:これらのリソースは DataWorks で直接編集されます。
- ファイルリソース:ファイルリソースを作成する際、ビッグファイルを選択する必要があります。ローカルリソースファイルもアップロードできます。

リソースインスタンスの作成

左側のナビゲーションバーで [Manual Business Flow] をクリックし、[Create Business Flow] を選択します。



2. [Resource] を右クリックし、[Create Resource] > [jar] を選択します。



 [Create Resource] ダイアログボックスが表示されます。 名前付け規則に沿って、リソース 名を入力し、リソースタイプを jar に設定します。アップロードするローカル jar パッケージ を選択します。 [Submit] をクリックし、開発環境へパッケージを送信します。

Create Resource				×
* Resource Name :	testJAR.jar			
Destination Folder :				
Resource Type :	JAR	~		
	✓ Upload to ODPS The resource will also be uploaded to ODPS.			
File :	Upload			
		ОК	Cancel	

道注:

- ODPS クライアントへこの jar パッケージがアップロードされた後、[Uploaded as the ODPS resource] の選択を解除する必要があります。このアップロードでは、リソースも ODPS へアップロードされます。そうでない場合、アップロードプロセス中にエラーが報告されます。
- ・ リソース名はアップロードするファイル名と同じである必要はありません。
- ・リソース名の名前付け規則:1から128文字以内の文字列で、文字、数字、アンダースコア、ドットを含みます。名前は大文字と小文字が区別されます。リソースが jar リソース である場合、拡張子は.jar です。

4. [Submit] をクリックし、リソースを開発スケジューリングサーバーへ送信します。

Upload Resource	
Saved Files :	ip2region.jar
Unique Resource Identifier :	OSS-KEY-l60u5o1g7t3g9uuim6j6polz
	Upload to ODPS The resource will also be uploaded to ODPS.
Re-upload :	Upload

5. ノードタスクを解放します。

操作に関する詳細情報は、「#unique_40」を参照してください。

1.7.3 関数

UDF **の登録**

MaxCompute では **UDF** をサポートしています。 詳細については、「*UDF* の概要」をご参照く ださい。

DataWorks では、add function という **ODPS** コマンドラインを置換するために関数を登録 する視覚的な **GUI** が提供されています。

現在、Python と Java API が UDF の実装をサポートしています。 UDF プログラムをコンパイ ルするには、「リソースの追加」を参考に UDF をアップロードし、UDF を登録します。

UDF **の登録手順**

 左側のナビゲーションウィンドウで [Manual Business Flow] をクリックし、[Create Business Flow] を選択します。



2. オフライン Java 環境で、プログラムを編集し、プログラムを jar パッケージへ圧縮し、jar リソースを作成します。そしてプログラムを送信し解放します。

別の方法としては、**Python** リソースを作成し、**Python** コードをコンパイルし保存します。 そしてコードを送信し解放します。 詳細については、「リソースの作成」 をご参照ください。

3. [Function] > [Create Function] を選択し、新しい関数の設定を入力し、[Submit] をク リックします。

Create Function			×
Function Name :	testFunction		
Destination Folder :			
	l	Submit	Cancel

4. 関数の設定を編集します。

Dete Developn 鬼 🗟 🗗 C 🕀 🕁	fo testfunction X
> Solution 88	Revisite Devisite
✓ Business Flow SS	registry Function
👻 🏯 base_cdp	Function Name : testFunction
✓ Cata Integration	* Class Norre : set
• 📴 write,result MeRCE 08-31 16:	
✓ m Data Development	* Resources : test.JAR jar
 insert_data Mel032 00-31153 	
• 👽 start Mel(22) 08-31 15:58	Lescrysten :
• 🖬 wedMR Mir/022 09-02-23-50	
• 🕞 168/SHELL 11-(102) 09-03 00	
• 🖆 testSQLComponent MolDE 0	Command Format:
• (v) tearVirtual Mettel: 09-03-00:2	Busentes
Y 🔝 Function	Paralleling (
• 🕋 testfunction Meltilit (🔂	
👻 🧸 workshop	
🛩 🛄 Data Integration	
• 🖂 fig_syne Melltill 09-02-17-26	
🖸 nda_sync: detaworka_demo203	
> 👩 Data Development	
> 🗾 Resource	

- Class Name: UDF を実装するメインクラスの名前です。 リソースが Python の場合、典型的な書き込みスタイルは、" Python リソース名.クラス名" (リソース名に py は不要です)。
- Resources: 第二段階のリソース名です。複数のリソースがある場合は、コンマで区切ります。
- ・ Description: UDF の説明です。 オプションです。
- 5. ジョブを送信します。

設定が完了したら、ページの左上隅にある[Save] をクリックする、または ctrl + S を押して ノードを開発環境へ送信 (ロックの解除) します。

6. ノードタスクを解放します。

操作に関する詳細は、「#unique_40」をご参照ください。

1.7.4 テーブル

テーブルの作成

1. [Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. [Table] を右クリックし、[Create Table] を選択します。



3. 基本属性を設定します。

- 111	Data Developn 鬼 🗟 📑 Ċ 🕀	e	🗰 gregrg	× Ja testJA	Rjar 🗙 🖪 tes	Function ×					
-			DOL Mode								
٠	> Solution	88									
R	✓ Business Flow	88			Table Name	gregrg					
Ĥ	> 🚠 base_cdp				Business Process	workshop					
	> 🚠 workshop										
Ľ			Basics								
				Table Alias :							
R				Level 1 Topic :	Select		Level 2 Topic	Select		Create Tonic	C
12											
-				Description :							
T											
			Physical Mod								
				_							
					Partition :	Partitioned	d Table 💿 Non-Partitioned	Table Life Cyc	le:		
				Table Level :	Select		Table Category	Select			C
			Table Structu								
-			Add Field	Move Up	Move Down						

- · Chinese Name: 作成するテーブルの中国語名です。
- ・ Level-1 Topic: 作成するテーブルの 第一階層のフォルダ名です。
- ・ Level-2 Topic: 作成するテーブルの第二階層フォルダ名です。
- · Description: 作成するテーブルの説明です。
- [Create Topic] をクリックします。表示された [Topic Management] ページで、第1
 階層と第2階層を作成します。

=				
영문 Configuration Center				
Project Configuration	Topic Enter	Venet Topic (Level 1 Topic by Default)	A.33	
Templates				
Theme Management	+ one_level	ungten	2018-09-03 13:49:41	
Table Levels				
Beckup and Restore				

4. DDL モードでテーブルを作成します。

[DDL Mode] をクリックします。表示されたダイアログボックスで、標準テーブルの作成文 を入力します。

	DDL Mod	e			×	
_						
Le						
ſ						
Model			Generate Table	Structure	Cancel	
Т	able Level :	Select	Table Category :	Select		

テーブル作成の SQL 文を編集した後、[Generate Table Structure] をクリックします。 [Basic Attributes] 領域、[Physical Model Design] 領域、[Table Structure Design] 領 域内の情報が自動的に入力されます。 5. GUI でテーブルを作成します。

DDL モードでのテーブル作成が難しい場合には、次の設定を行うと、GUI でテーブルを作成できます。

- · [Physical model design] 領域
 - Partition Type:「Partitioned Table」または「Non-partitioned Table」を設定 できます。
 - Life Cycle: MaxCompute のライフサイクル機能です。Life Cycle (単位:日) で指定 した期間内にアップロードされていないテーブル (またはパーティション) 内のデータは 消去されます。
 - Level: 「DW」、「ODS」、「RPT」のいずれかを設定できます。
 - Physical Category: 「Basic Business Layer」、「Advanced Business Layer」、「Other」のいずれかを設定できます。 [Create Level] をクリックします。 表示された [Level Management] ページで階層を作成します。
- · [Table structure design] 領域
 - English Field Name: フィールドの英語名です。文字、数字、アンダースコア(_)を使用できます。
 - Chinese Name: フィールドの略称された中国名
 - Field Type: MaxCompute のデータタ型です。「String」、「Bigint」、「Double」、「Datetime」、「Boolean」のみ使用できます。
 - Description: フィールドの詳しい説明です。
 - Primary Key: フィールドが主キーか、結合条件の主キーの場合に選択します。
 - 新しいフィールドに列を追加するには、[Add Field] をクリックします。
 - 作成したフィールドを削除するには、[Delete Field] をクリックします。

注:

作成したテーブルからフィールドを削除した後、そのテーブルを再度送信する場合、現 行テーブルを破棄して、同じ名前で新しいテーブルを作成する必要があります。 この 操作は、本番環境で行うことはできません。

- 作成するテーブルのフィールドの順番を調整するには、[Move Up] をクリックします。 ただし、作成したテーブルのフィールドの順番を調整する場合、現行テーブルを破棄し て、同じ名前で新しいテーブルを作成する必要があります。 この操作は、本番環境で行 うことはできません。

- [Move Down] の操作は、[Move Up] の場合と同じです。
- 現行テーブルにパーティションを作成するには、[Add Partition] をクリックします。
 作成したテーブルにパーティションを追加する場合、現行テーブルを破棄して、同じ名前で新しいテーブルを作成する必要があります。この操作は、本番環境で行うことはできません。
- パーティションを削除するには、[Delete Partition] をクリックします。作成したテー ブルからパーティションを削除する場合、現行テーブルを破棄して、同じ名前で新しい テーブルを作成する必要があります。この操作は、本番環境で行うことはできません。
- Action:新しいフィールドの追加、フィールドの削除、および他の属性の編集を確認します。

他の属性には、データ品質に関連する情報が含まれます。これはシステムに対して、検 証ロジックを生成するために提供されています。 データプロファイル、SQL スキャン、 テストルールの生成などのシナリオで使用されます。

- 0 Allowed: 選択するとフィールド値をゼロにすることができます。 このオプション は、Bigint 型と Double 型のフィールドにのみ適用されます。
- Negative Value Allowed: 選択すると、フィールド値を負の数にすることができま す。 このオプションは、Bigint 型と Double 型のフィールドにのみ適用されます。
- Security Level:「[Non-sensitive」、「Sensitive」、「Confidential」のいずれかに 設定できます。

C: 顧客データ、B: 企業データ、S: ビジネスデータ C1-C2、B1、S1 は 非機密 (non-sensitive) データです。 C3、B2-B4、S2、S3 は機密 (sensitive) データです。 C4、S4、B4 は極秘 (Confidential) データです。

- Unit: 金額の単位で、ドルやセントを選択できます。 このオプションは金額に関連の ないフィールドには必要ありません。
- Lookup Table Name/Key Value: 会員タイプやステータスといった列挙型フィー ルドに適用されます。 このフィールドに対応した辞書テーブル (またはディメンショ ンテーブル) の名前を入力します。 たとえば、会員ステータスに対応した辞書テーブ ル名は、"dim_user_status" です。 グローバルユニークな辞書テーブルを使用する

場合、辞書テーブル内のフィールドに対応する key_type を入力します。 たとえば、 会員ステータスの対応キー値は、"TAOBAO_USER_STATUS" です。

- Value Range: 現行フィールドの最大値と最小値です。 Bigint 型 と Double 型の フィールドにのみに適用されます。
- Regular Expression Verification: 現行フィールドで使用する正規表現です。た とえば、フィールドが携帯電話番号で、正規表現 (またはさらに厳格な制限) で 11 桁 の数字に制限することができます。
- Maximum Length: フィールド値の最大文字数です。 String 型のフィールドにの み適用されます。
- Date Precision: 日付の精度です。「Hour」、「Day」、「Month」、「others 」のいずれかに設定できます。たとえば、フィールド値が 2014-08-01であっても (精度は「Day」のように見えます)、月次サマリーテーブル "month_id" の精度は 「Month」です。Datetime 型 または String 型の日付値に適用されます。
- Date Format: String 型の日付値にのみ適用されます。 フィールドに実際に格納さ れる日付値の形式は、yyyy-mm-dd hh:mm:ss のようになります。
- KV Primary Separator/Secondary Separator: キー値ペアが組み合わされた大 規模なフィールド (String型) に適用されます。たとえば、プロダクト拡張属性に " key1:value1;key2:value2;key3:value3;..."といった値がある場合、セミコロン (;) はフィールドの主となるセパレーターで、キー値ペアを区切ります。コロン (:) は 二次的なセパレーターで、キー値ペアのキーと値を区切ります。
- Partition Field Design: このオプションは、[Physical Model Desing] 領域の [
 Partition Type] が「Partitioned Table」に設定されている場合にのみ表示されます。
- · Field Type: すべてのフィールドに対して、String 型を使用することを推奨します。
- Date Partition Format: パーティションフィールドが日付 (データ型は String) の場合、
 「yyyymmmdd」といった日付形式を選択または入力します。
- ・ Date Partition Granularity: たとえば、「Day」、「Month」、「Hour」です。

テーブルを送信します。

- テーブル構造情報を編集した後、新規テーブルを開発環境および本番環境へ送信します。
- [Load from Development Environment] をクリックします。テーブルが開発環境に送信 されると、このボタンはハイライトされます。このボタンをクリックすると、開発環境で作成 したテーブルの情報が、現在のページ情報に上書きされます。
- [Submit to Development Environment] をクリックすると、現在の編集ページで必要な 項目がすべて完全に設定されているかどうか確認されます。未設定の項目がある場合、テー ブルの送信を禁止するアラームが報告されます。

- ・ [Load from Production Environment] をクリックすると、本番環境へ送信したテーブルの詳細情報が、現在のページ情報に上書きされます。
- ・ [Create in Production Environment] をクリックすると、本番環境のプロジェクトにテー ブルが作成されます。

1.8 マニュアルタスクのノードタイプ

1.8.1 仮想ノード

仮想ノードは、データを生成しない制御ノードです。 通常、ワークフローのノード計画全体に対 するルートノードとして使用します。

仮想ノードタスクの作成

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. 仮想ノードを作成します。 [Data Development] を右クリックし、[Create Data Development Node] > [Virtual Node] の順に選択します。

0	Enter a file or creator name	\ [r 💿 🗉 🔊
*	> Solution		✓ Data Integration Develop
R	➤ Business Flow		
Ĥ	✓ ♣ base_cdp	»	Di Data Sync
	> 😑 Data Integration		✓ Data Development
	> VD Dε	mentNo	de ID > ODPS SQL
#	> III Ta Create Folder		Shell Create Data DevelopmentNode ID
	> 🧭 Re Board		ODPS MR
E.	🔉 🔂 Fu 🛛 Reference Compone	Virtual Node	
£.	> 🔚 Algorithm		
	> 🚳 control		Py F SQL Component Node
Û	> A works		
	- WORS		Node
	> 📇 workshop		Mr OPEN MR

3. ノードタイプを [Virtual Node] に設定し、ノード名を入力します。ターゲットフォルダを選択し、[Submit] をクリックします。

Create Node		×
Node Type :	Virtual Node 🗸 🗸 🗸	
Node Name :	testVirtual	
Destination Folder :		
	Submit	Cancel

4. ノードコードの編集: 仮想ノードでは、コードを編集する必要はありません。
5. ノードスケジューリングの設定

ノードタスク編集エリアの右側にある [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。 詳細は、「スケジューリングの設定」 をご参照ください。

6. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、 または、[Ctrl] + [S] を押してノードを環境開発へ送信 (およびロックを解除) します。

7. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

8. 本番環境でテストします。

操作の詳細については、「#unique_46」をご参照ください。

1.8.2 SQL コンポーネントノード

手順

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. SQL コンポーネントノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [SQL Component Node] の順に選択します。

0	Enter a file or creator name	V.	
*	> Solution		 Data Integration
R	➤ Business Flow		
Ĥ	✓ ♣ base_cdp	»	Di Data Sync
	> 😑 Data Integration		✓ Data Development
2	> Create Date Develop	mentNo	de ID > ODPS SOL
Ħ	Ta Create Folder Market Report		Shell Create Data DevelopmentNode ID
R	> 🔁 Fu Reference Compone	ent	Virtual Node
£×	> 📜 Algorithm		VI V PyODPS
	> 🜀 control		Py F SQL Component Node
Û	> 嚞 works		OPEN MR 5 이 North
	> 🎝 workshop		

- **3.** 開発効率を向上させるため、データタスクの開発者はプロジェクトメンバーとテナントメン バーによって提供されたコンポーネントを使い、データ処理ノードを作成します。
 - ・ ローカルプロジェクトのメンバーが作成したコンポーネントは、[Project Components]
 にあります。
 - ・テナントメンバーが作成したコンポーネントは、[Public Components] にあります。

ノードを作成する際、ノードタイプを「SQL component node」に設定し、ノード名を指 定します。



選択したコンポーネントのパラメーターを指定します。

ш	Tables	C C	Û	myComponent 🔵	🖆 testSQL0	Component	× 💵 te	estMR		🌆 testJAR.jar		info_d x	Se dw_user_info_all_d ×	Sq ods_log_	info_d ×	
			Ľ	" 🕆 F		00		22								
*	🗸 🛅 Tables			1 SQL com	ponent mod	del						x				
R	🛩 🛅 Others											* Owner :	wangdan			
ė.	m bank_data											Description :				
R	bank_data1															
	odps_result			8 insert over 9 partition	rwrite tal	ble @@{my	_output	_table}				Input Parameters(?)				
=	ds_log_info_d			10 select	(03- \$101	tuate;)						Parameter Name	mycompent	* Type :	String	
5	🗏 otheretion															
52	H ivironmen			13 @@{my_ 14 where ca	input_tabl tegory in	le} ('@@{≡y_	input_p	aramete	r1}',	, '00{my_in	put_	Description :	default value			
ŧ	result_table				bstr(pt, 1	1, 8) in	('\${biz	date}')								
												Default Value :	bank_data			
												Output Parameters(?)				
													a		0 -1	
									不			* Parameter Name	4	* Type :	String	
									5.7			Description				
								,	2.71			Description :				
۵												Default Value :	4			

パラメーター名を入力し、パラメータータイプを「Table」または「String」に設定します。 3つの "get_top_n" パラメーターを順番に指定します。

テーブルタイプ (test_project.test_table) のパラメーターに対して次の入力テーブルを指定 します。 4. ノードスケジュールの設定

ノードタスク編集エリアの右側で [Schedule Configuration] をクリックし、[node

scheduling configuration] ページへ移動します。詳細は、「スケジューリング設定」をご 参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックを解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_46」をご参照ください。

SQL コンポーネントノードのバージョンのアップグレード

コンポーネントの開発者が新しいバージョンをリリースすると、コンポーネントユーザーはイン スタンスで使用する既存のコンポーネントを最新版のコンポーネントにアップグレードするかど うかを選択できます。

コンポーネントのバージョンメカニズムにより、開発者は継続的にコンポーネントをアップグ レードできます。また、コンポーネントユーザーは、コンポーネントのアップグレードによって 改善されるプロセスの実行効率と最適化されたビジネス効果を継続的に享受できます。

たとえば、ユーザー A は、ユーザー C が開発した v1.0 コンポーネントを使用しており、コン ポーネントオーナー C がコンポーネントを V2.0 にアップグレードしたとします。 アップグレー ド後もユーザー A は、v1.0 コンポーネントを使用し続けることができますが、アップグレード のリマインダーを受け取ります。 ユーザー A は旧コードと新コードを比較し、旧バージョンより も新しいバージョンのビジネス効果の方が高いと感じた場合、最新版へアップグレードするかど うかを決定できます。

コンポーネントテンプレートに基づいて開発された SQL コンポーネントノードをアップグレー ドするには、[Upgrade] を選択して、SQL コンポーネントノードのパラメーター設定が新しい バージョンでも有効かどうかを確認するだけで済みます。新しいバージョンのコンポーネントの 手順に沿って調整を行い、通常の SQL コンポーネントノードと同じように送信しリリースしま す。

インターフェイスの機能

Data Developn 🖉 📑 🖓 😷 🕀	¢	🔯 Data Development x 🖒 myComponent . 🌒 🏠 teatSQLComponent x 📾 teatMR 🛛 x		≡
Enter a file or c Code Search				
> Solution		3 2 3 4 5 6 7 8 ×		
		Component 2 Select a component v Boats Code Version Inget Parameters		
✓ ▲ bese,cdp		1 Note		
Deta Integration		Output Parameters@		
Y 🔯 Data Development		Nove		loh a
• 🔄 insert_data Mr(052-004				dule
• 🕥 start Mel(02) 08-31 15.5				
• 🔤 testMR Mel022 09-02.2				R.
• D textSHELL Meltic 09-0	3 0Q 6		11	
• 🗋 test5QLComponent Mel				ship
• in testVirtual MeRCE 09-0				
> 🔲 Table			12	Versi
> 🔀 Resource				

インターフェイスの機能は次のとおりです。

No.	機能	説明
1	Save	クリックして、現在のコンポーネントの設定を保存しま す。
2	Submit	クリックして、現在のコンポーネントを開発環境へ送信 します。
3	Submit and Unlock	クリックして、現在のノードを送信し、コードを編集で きるようにロックを解除します。
4	Steallock Edit	現在のコンポーネントのオーナーではない場合、クリッ クし、ノードを steallock 編集します。
5	Run	クリックし、開発環境内でローカルでコンポーネントを 実行します。
6	Advanced Run (with Parameters)	クリックして、コードに対して設定されたパラメーター を使って現在のノードのコードを実行します。
		注: Advanced Run は、Shell ノードでは利用できません。
7	Stop Run	クリックし、実行中のコンポーネントを停止します。
8	Re-load	クリックして、インターフェイスを更新し、最後に保存 した状態に戻します。 保存されていないコンテンツは失 われます。
		注: 設定センターでキャッシュを有効にしている場合、イ ンターフェイスが更新された後、コードはキャッシュ されますが保存はされません。この場合、必要なバー ジョンを選択します。

No.	機能	説明
9	Parameter Settings	クリックし、コンポーネント情報、入力パラメーター設 定、出力パラメーター設定を表示します。
10	Attributes	クリックして、ノードのオーナー、説明、パラメー ター、リソースグループを設定します。
11	Kinship	SQL コンポーネントノード間の親子関係マップや、各 SQL コンポーネントノードの内部親子関係マップを表示 します。
12	Version	現在のコンポーネントの送信および発行履歴を表示しま す。

1.8.3 ODPS MR ノード

MaxCompute では、MapReduce プログラミング API をサポートします。 MapReduce によ り提供される Java API を使って、MaxCompute でデータを処理するための MapReduce プロ グラムを作成することができます。 ODPS MR ノードを作成し、タスクスケジューリングで使用 します。

ODPS MR の編集方法や使用方法は、**MaxCompute** ドキュメントの「*WordCount* の例」に記載 されている例をご参照ください。

ODPS MR ノードを使用するためには、使用するリソースを最初にアップロードしてリリースします。そして ODPS MR ノードを作成します。

リソースインスタンスの作成

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. [Resource] を右クリックし、[Create Resource] > [jar] の順に選択します。



3. 命名規則に従い、[Create Resource] にリソース名を入力し、リソースタイプを「jar」に設定します。アップロードするローカル jar パッケージを選択します。

Create Resource				×
* Resource Name :	testJAR.jar			
Destination Folder :				
Resource Type :	JAR	~		
	Upload to ODPS The resource will also be uploaded to ODPS.			
File :	Upload			
		ОК	Cancel	

注:

- この jar パッケージが ODPS クライアントへアップロードされた後、[Uploaded as the ODPS resource] の選択を解除する必要があります。このアップロードでは、リソースも ODPS ヘアップロードされます。そうでない場合、アップロードプロセス中にエラーが報告されます。
- ・リソース名は、アップロードされるファイル名と同じにする必要はありません。
- ・リソース名の命名規則:1 文字以上 128 文字以内の文字列で、英数字、アンダースコア、 ドットを使用できます。名前は大文字と小文字が区別されません。リソースが jar リソー スの場合、拡張子は .jar です。リソースが Python リソースの場合、拡張子は .py で す。

4. [Submit] をクリックして、リソースを開発スケジューリングサーバーへ送信します。

Upload Resource	
Saved Files :	ip2region.jar
Unique Resource Identifier :	OSS-KEY-l60u5o1g7t3g9uuim6j6polz
	✓ Upload to ODPS The resource will also be uploaded to ODPS.
Re-upload :	Upload

5. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

ODPS MR ノードの作成

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. ODPS MR ノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [ODPS MR] の順に選択します。



3. ノードコードを編集します。新しい **ODPS MR** ノードをダブルクリックし、次のインターフェ イスに移動します。

Data Developn 온 🛱 📮 Ċ 🕀 🗗	🔉 ip2r	egion.jar 🗙	w test	AR 🔴	🕢 数据开发		vi workshop_start ×	\boxed{sq} create_table_ddl ×	testMR
Enter a file or creator name		B 0	1	÷ 0) :				
 Function Algorithm 		odps r author	r time:20		7 16:17:18	•••••			
> 🖸 control		*****	******	•••••	*********	******	*****		
> 🏯 works									
👻 🚣 workshop									
> 🚞 Data Integration									
Y 📅 Data Development									
Sq create_table_ddl dataworks_									
• In testMR Melocked 09-171									
🔤 dw_user_info_al_d datawork									
🔄 ods_log_info_d deterworks_dr									
• Sq rpt_user_info_d Mellocked 0									
vi workshop_start Melocked 0									
> 🔲 Table									
🗸 🔁 Resource									
ip2region.jar Mellocked 09-1									
• 🗈 test.JAR.jør dataworks_demo									

ノードコードの編集例

jar -resources base_test.jar -classpath ./base_test.jar com.taobao. edp.odps.brandnormalize.Word.NormalizeWordAll

コードの説明は次のとおりです。

- ・ -resources base_test.jar:参照先の jar リソースのファイル名を示します。
- -classpath: jar パッケージのパスです。参照リソースを右クリックし、このパスを取得します。

新しい ODPS MR ノードをダブルクリックして、ODPS MR ノードインターフェイスに移 動し、jar リソースを入力します。

 com.taobao.edp.odps.brandnormalize.Word.NormalizeWordAll:実行時に呼び 出される jar パッケージのメインクラスを示します。jar パッケージのメインクラス名と 同じにする必要があります。

1 つの MR から複数の jar リソースを呼び出す場合、クラスパスは-classpath ./xxxx1. jar,./xxxx2.jar のとおりに記述する必要があります。2 つのパスはコンマで区切ります。 4. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリング設定」をご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctr]l + [S] を 押して ノードを開発環境へ送信 (およびロックを解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_46」をご参照ください。

1.8.4 マニュアルデータ統合ノード

現在、データ統合 (data Integration) タスクで使用できるデータソースは、MySQL、DRDS、 SQL Server、PostgreSQL、Oracle、MongoDB、DB2、Table Store、OTSStream、 OSS、FTP、Hbase、LogHub、HDFS、および Stream です。 サポートされているデータ ソースについての詳細は、「#unique_17」をご参照ください。



1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで、[Manual Business Flow] 、[Create Business Flow] の順に選択します。



2. データ統合ノードを作成します。

[Data Integration] を右クリックし、[Create Data Data Integration Node] > [Data Integration] の順に選択します。

	Enter a file or creator name	
*	> Solution	Data Integration Deve
R	➤ Business Flow	
8	✓ 🗸 base_cdp	Di Data Sync
	> 🔁 Data Integration	
2	> 🕢 Data Developm	Create Data IntegrationNode ID > Data Sync
) 🔲 Table	Create Folder
#		Board
_	> 🧭 Resource	
B	> 🔂 Function	Mr ODPS MR
£×	> 🔚 Algorithm	Vi Virtual Node
	> 🧭 control	Py PyODPS
Ū	> 嚞 works	SQL Component Node
	> 📇 workshop	Mr OPEN MR

3. 統合タスクを設定します。

ソーステーブル名とターゲットテーブル名を入力すると、タスクの簡易設定が完了します。 テーブル名を入力すると、そのテーブル名と一致するオブジェクトのリストが自動的に表示さ れます (現在、完全にテーブル名が一致するオブジェクトのみが表示されるため、 正しく完全 なテーブル名を入力する必要があります)。 現在の統合センターではサポートされていないオ ブジェクトは [Not supported] と表示されます。 オブジェクトにマウスを移動します。 デー タベース、IP アドレス、テーブルのオーナーといったオブジェクトの詳細情報が自動的に表 示されます。 適切なテーブルオブジェクトを選択するのに役立ちます。 オブジェクトを選択 したら、そのオブジェクトをクリックします。 列の情報が自動的に入力されます。 移動、削 除、追加といった列の編集を行うことができます。

a. 統合テーブルを設定します。

	Data Develope A 🕞 🖓 🗘 🗘 🕁	🕒 skee, sine 🕣				
-	Enter a file or creator name	Ľ ⊙ ⊡				
*	> Solution 🔡	(C) Party and				
R	✓ Business Flow	C Distanti				
8	 M base, rdp Data Integration 		The data sources can be default data source	s or data sources created by you. Click her	a to check the supported data source types.	
	• 🔀 wike,weak 16-(53) 08-31 1	Dete Source :	00PS v odps.fest v	Data Source :	MySQL v rds_workshop_log v	0
•	 Data Development Table 	* Table :	reul_able	• Table:	ingion_day_stat	
R	> 🔯 Resource	Partition :	None	Statements Run :		0
83	Function Backing	Compression	😑 Disable 🔿 Enable	Before Import		
	> 🧰 control > 🏯 works	Consider Empty String as Null	🔵 Yes 🔿 No	Statements Run:		Ø
	> 👗 workshop			Autor Industri		
				* Solution to: Duplicate Primary	INSERT INTO ~	
				Кауа		
		Mapping	Source Table		stination Table	
			Field Tore (B)		Feld Too	Map of the same

b. データソースを編集します。

通常、必要がない限り、ソーステーブルのコンテンツを編集する必要はありません。

- ・新しい列を挿入するには、列の右側にある [Insert] をクリックします。
- ・列を削除するには、、列の右側にある [Delete] をクリックします。
- c. データの統合先の情報を編集します。

通常、必要がない限り統合先のテーブルのフィールド情報を編集する必要はありません。

統合先が ODPS テーブルの場合、列を削除することはできません。 統合タスクの設定で は、ソーステーブルのフィールド設定は、フィールド名ごとではなくページごとに、統合 先のテーブル設定と1対1で合致しています。

- d. 増分統合とフル統合
 - ・ 増分統合のシャード形式: ds=\${bizdate}
 - ・フル統合のシャード形式: ds=*

三注:

複数のシャードを同期する必要がある場合、統合センターは簡易正規表現をサポートしま す。

- ・たとえば、複数のシャードを同期する必要があるが、正規表現の記述が難しい場合は、
 メソッドds=20180312 | ds=20180313 | ds=20180314; を使用します。
- シャードを同じ範囲内で同期する必要がある場合、統合センターは、/*query*/ds>=
 20180313 and ds<20180315; といった拡張構文をサポートします。このメソッド
 を使用する場合、/query/を追加します。
- ・ 変数 bizdate は、パラメーター-p"-Dbizdate=\$bizdate -Denv_path=\$
 env_path -Dhour=\$hour"で定義する必要があります。 たとえば、pt=\${selfVar
 } といった変数をカスタマイズする場合、パラメーターの変数を-p"-Dbizdate=\$
 bizdate -Denv_path=\$env_path -Dhour=\$hour -DselfVar=xxxx"のように
 定義します。
- e. フィールドのマッピング

フィールドは、フィールド名やフィールドタイプではなく、ソーステーブルと統合先のテー ブルのフィールド位置に基づいてマッピングされます。

••••	1 8 0	ò @				
	Field	Туре 🥝	3	Field	Туре	Map of the same name
	education	STRING	•	💿 bizdete	DATE	Enable Same-Line Mapping
	num	BIGINT	•	💿 region	VARCHAR	
	Add +				BIGINT	
					BIGINT	
				browse_size	BIGINT	
03 Channel						
You	can control the data synch	ronization process th	rough the transmission rate and the number of allo	owed dirty data records. See	data synchronization docu	ments.
	umo.			0		
* Numbe	r of Concurrent Jobs : 2		0			
	* Transmission Rate : 💿	Unlimited 🔾 Limite	ed			
н	there are more than : Mo task	nimum r@ber of di kends.		dirty data records, the		
Ta	sk's Resource Group : De	fault resource group	Ŷ]		
注 注:	:					

ソーステーブルが ODPS テーブルの場合、データの統合中にフィールドを追加することは できません。 ソーステーブルが ODPS テーブル以外の場合、データの統合中にフィール ドを追加することができます。

f. トンネル制御

トンネル制御機能は、統合タスクを選択する際の速度やエラー率を制御するために使用します。

- ・ DMU: データ統合中に消費されるリソース (CPU、メモリ、ネットワーク帯域幅など) を測定するデータ移行単位
- Concurrent job count: データ統合タスクでデータ記憶媒体へデータの書き込み、
 データ記憶媒体からのデータの読み込みを同時に行うために使用されるスレッドの最大数
- integration speed: 統合タスクの最高速度
- Maximum error count: ダーティデータ量を制御するために使用します。ソース元 テーブルのフィールドタイプが統合先のフィールドタイプと一致しない場合、同期デー タ量に基づいてユーザーが設定します。許容されるダーティデータカウントの最大数を 指定します。「0」に設定すると、ダーティデータは一切許容されません。指定してい ない場合、ダーティデータは許容されます。
- Task resource group: 現在の統合ノードがあるリソースグループを選択するには、 データ統合ページでリソースグループを追加または編集します。
- 4. ノードスケジューリングの設定

ノードタスク編集エリアの右側にある [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」をご参照ください。

5. ノードタスクを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押してノードを開発環境へ送信 (およびロックを解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_20」をご参照ください。

1.8.5 PyODPS ノード

DataWorks では、**PyODPS** タスクも実行することができ、**MaxCompute** の **Python** SDK と統合されます。 **Python** コードを直接編集して、**DataWorks** の **PyODPS** ノードで **MaxCompute** を操作することができます。

PyODPS ノードの作成

MaxCompute には、MaxCompute の操作に使用できる Python SDK が実装されています。

PyODPS ノードを作成するには、次の手順に従います。

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで[Manual Business Flow] 、[Create Business Flow] の順にを選択します。



2. PyODPS ノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [PyODPS] の順に選択します。

ω	Enter a file or creator name	V	f) 💿 🔍 🕅
*	> Solution		✓ Data Integration Develop
R	✓ Business Flow		
Ĥ	✓ ♣ base_cdp	»	Di Data Sync
	> 😑 Data Integration		 Data Development
Ž	> 🕖 Da 🕺	mentNo	de ID > ODPS SOL
#	 > III Ta Create Data Develop Create Folder > Ø Re Board 	Shell Create Data DevelopmentNode ID ODPS MR	
12	> 🔁 Fu 🛛 Reference Compone	ent	Virtual Node
∱×	> 🔚 Algorithm	VI V PyODPS	
Û			OPEN MR
	Works		5 Node
	> 🚠 workshop		Mr OPEN MR

- 3. PyODPS ノードを編集します。
 - a. ODPS ポータル

DataWorks では、PyODPS ノードには、ODPS エントリであるグローバル変数 "odps" または "o" が含まれています。手動で ODPS エントリを定義する必要はありません。

```
print(odps.exist_table('PyODPS_iris'))
```

b. SQL 文を実行します。

PyODPS では、**ODPS SQL** クエリをサポートしており、実行結果を読み取ることができま す。 **execute_sql** または **run_sql** メソッドのリターン値は実行中のインスタンスです。

注: ODPS コンソールで実行できるコマンドのすべてが、ODPS で受け入れられる SQL 文ということではありません。DDL 文や DML 文以外を呼び出すには、別のメソッ ドを使用する必要があります。たとえば、GRANT 文や REVOKE 文を呼び出すに は、"run_security_query" メソッドを使用します。PAI コマンドを呼び出すに

は、**"run_xflow"** または **"execute_xflow"** メソッドを使用します。

o.execute_sql('select * from dual') # Run the SQL statements in synchronous mode. Blocking continues until execution of the SQL statement is completed. instance = o.runsql('select * from dual') # Run the SQL statements in asynchronous mode. print(instance.getlogview_address()) # Obtain the logview address . instance.waitforsuccess() # Blocking continues until execution of the SQL statement is completed.

c. ランタイムパラメーターを設定します。

ランタイムパラメーターの設定が必要になる場合があります。 パラメータータイプ dict を 使って、ヒントパラメーターを設定します。

o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper .split.size': 16})

sql.setings をグローバル設定に追加すると、関連するランタイムパラメーターが、各 **running.python** に追加されます。

from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris') # "hints" is added
based on the global configuration.

d. SQL 文の実行結果を読み取ります。

SQL 文を実行するインスタンスでは、open_reader 操作が直接実行されます。 あるケー スでは、構造化されたデータが SQL 文の実行結果として返されます。

with odps.execute_sql('select * from dual').open_reader() as reader: for record in reader: # Process each record.

別のケースでは、SQL 文で desc が実行される場合があります。 この場合、オリジナルの SQL 文の実行結果は、reader.raw 属性を使って取得されます。

```
with odps.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

_____注:

ユーザー定義のスケジューリングパラメーターは、データ開発に使用されます。 PyODPS ノードがページ上で直接トリガーされる場合、時間を明確に指定する必要があります。 PyODPS ノードの時間は、SQL ノードの時間のように直接置換することができません。 4. ノードスケジューリングの設定

ノードタスク編集エリアの右側で [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」をご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅の [Save] をクリックするか、または [Ctr]l + [S] を押し てノードを開発環境へ送信 (およびロックを解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

7. 本番環境でテストします。

操作に関する詳細は、「#unique_46」をご参照ください。

1.8.6 ODPS SQL **ノード**

ODPS SQL の構文は SQL 構文と似ています。ODPS SQL 構文は、データ量は膨大 (TB 級) だ が、リアルタイム要件が高くない分散シナリオに適用されます。 スループット重視の OLAP ア プリケーションです。 ジョブの準備から送信までのプロセスが完了するのに長時間がかかるた め、ODPS SQL は数万単位でのトランザクションを処理する必要のあるビジネスに対して推奨さ れます。

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow]、[Create Business Flow] の順に選択します。



2. ODPS SQL ノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [ODPS SQL] の順に選択します。

0	Enter a file or creator name	V	f) 💿 🔍 🕅
*	> Solution	88	✓ Data Integration Develop
	 Business Flow Base_cdp 	88 N	Di Data Sync
1	> 🚍 Data Integration		 Data Development
	> ひ D ∈ Create Data Develop	omentNo	ode ID > ODPS SQL
#	 Create Folder Re Board 		Shell Create Data DevelopmentNode ID ODPS MR
2	🔉 🔁 Fu 🛛 Reference Compone	ent	Virtual Node
£×	> 📜 Algorithm		VI V PyODPS
Ť	> 🧭 control		
	> 🗸 workshop		Node Mr OPEN MR

3. ノードコードを編集します。

SQL 構文に関する詳細については、「*MaxCompute SQL* 文」をご参照ください。

4. ノードスケジューリングの設定

ノードタスク編集領域の右側にある [Schedule] をクリックし、[node scheduling configuration] ページへ移動します。詳細は、「スケジューリングの設定」 をご参照ください。

5. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctrl] + [S] を 押して開発環境へノードを送信 (ロックを解除) します。

6. ノードタスクを発行します。

操作の詳細については、「リリース管理」をご参照ください。

7. 本番環境でテストします。

操作の詳細については、「#unique_46」をご参照ください。

1.8.7 SHELL ノード

SHELL タスクは、標準の SHELL 構文をサポートしていますが、対話型構文はサポートしてい ません。 SHELL タスクは、デフォルトリソースグループで実行されます。 IP アドレスやドメイ ン名にアクセスする場合、[Project Configuration] を選択し、IP アドレスまたはドメイン名 をホワイトリストへ追加します。

手順

1. ビジネスフローを作成します。

左側のナビゲーションウィンドウで [Manual Business Flow] 、[Manual Business Flow] の順に選択します。



2. SHELL ノードを作成します。

[Data Development] を右クリックし、[Create Data Development Node] > [SHELL] の 順に選択します。

0	Enter a file or creator name	∑	r 💿 🗉 🔊
*	> Solution		✓ Data Integration Develop
B	➤ Business Flow		
8	✓ ♣ base_cdp	»	Di Data Sync
	> 😑 Data Integration		✓ Data Development
	> VI DE	mentNo	de ID > ODPS SOL
Ħ	> III Ta Create Folder	Jinenuvo	Shell Create Data DevelopmentNode ID
	> 💋 Ke Board		ODPS MR
E.	> 🛃 Fu 🛛 Reference Compone	ent	Virtual Node
	> 📜 Algorithm		VI V PyODPS
			SQL Component Node
Ĥ			OPEN MR
	> 🚠 works		ے کے <mark>N</mark> ode
	> 🏝 workshop		Mr OPEN MR

3. ノードタイプを「SHELL」に設定し、ノード名を入力します。ターゲットフォルダを選択 し、[Submit] をクリックします。 4. ノードコードを編集します。

SHELL ノードコードの編集ページへ移動し、コードを編集します。

	Data Developn 🖉 🗟 📮 C 🕀 🕁	Sh testSHI	ELL × 🖂 1	estMR	× Ja	• "	e nie nas been createu —	Sa dw_user_info_al
673	Enter a file or creator name		🖫 🗗 🛛	6	\odot			
*	> Solution							
R	➤ Business Flow		##author:wan	gdan	******	*******	* * * * * * * * * * * * * * * * * * * *	******************
-	🗸 🏝 base_cdp		#create tim	e:2018-	09-03 0	0:02:13	*****	***************
B	> 🧮 Data Integration							
	✓							
	• Sq insert_data Me锁定 08-31 15							
47	• VI etert Ma総宗 08-31 15-58							

SHELL 文でシステムスケジューリングパラメーターを呼び出す場合、次のとおり SHELL 文 を記述します。

echo "\$1 \$2 \$3"

_____注:

Parameter 1 Parameter 2... 複数のパラメーターはスペースで区切ります。 システムスケ ジューリングパラメーターの使用に関する詳細は、「パラメーターの設定」をご参照くださ い。

5. ノードスケジューリングの設定

ノードタスク編集エリアの右側の [Schedule] をクリックし、[node scheduling

configuration] ページへ移動します。詳細は、「スケジューリングの設定」をご参照ください。

6. ノードを送信します。

設定が完了した後、ページの左上隅にある [Save] をクリックするか、または [Ctr]l + [S] を 押してノードを開発環境へ送信 (およびロックを解除) します。

7. ノードタスクをリリースします。

操作の詳細については、「リリース管理」をご参照ください。

8. 本番環境でテストします。

操作の詳細については、「#unique_46」をご参照ください。

使用例

SHELL を使ってデータベースへ接続します。

Alibaba Cloud でデータベースを構築し、リージョンが中国 (上海)の場合、次のホワイトリスト内の IP アドレスを使用してデータベースを開き、接続します。

10.152.69.0/24、10.153.136.0/24、10.143.32.0/24、120.27.160.26、10.46.67.156、120.27.160.81、10.46.64.81、121.43.110.160、10.117.39.238、121.43.112.137、10 .117.28.203、118.178.84.74、10.27.63.41、118.178.56.228、10.27.63.60、118.178. 59.233、10.27.63.38、118.178.142.154、10.27.63.15、100.64.0.0/8

🧾 注:

Alibaba Cloud でデータベースを構築したが、リージョンが中国 (上海) ではない場合、イ ンターネットを使用するか、またはデータベースと同じリージョンで ECS インスタンスをス ケジューリングリソースとして購入し、カスタムリソースグループで SHELL タスクを実行 することを推奨します。

 ・データベースがローカルで構築されている場合、インターネット接続を使い、上記のホワイト リスト内の IP アドレスでデータベースを開くことを推奨します。

道注:

カスタムリソースグループを使って SHELL タスクを実行する場合、カスタムリソースグ ループ内のマシンの IP アドレスを上記ホワイトリストに追加する必要があります。

1.9 マニュアルタスクパラメーターの設定

1.9.1 基本属性

次の図は、基本属性の設定インターフェイスを示しています。

Basics ⑦				
Node Name:	insert_data	Node ID:		
Node Type:	ODPS SQL	Owner:	10	
Description:				
Parameters:	bizdøte=Sbizdøte døtetime=S(yyyymmdd)			0

- Node Name: ワークフローノードを作成する際に入力するノード名です。ノード名を編集するには、ディレクトリツリーでノードを右クリックし、ショートカットメニューから [Rename]を選択します。
- ・ Node ID: タスクの送信時に生成される一意のノード ID で、編集はできません。
- ・ Node ID: タスクの送信時に生成される一意のノード ID で、編集はできません。

 Owner: ノードのオーナーです。デフォルトでは現在ログインしているユーザーが、新しく 作成したノードのオーナーになります。オーナーを変更するには、入力ボックスをクリックし て、オーナーの名前を入力するか、または他のユーザーを選択します。

注:

他のユーザーを選択する場合、そのユーザーは現行プロジェクトのメンバーである必要があ ります。

- · Description: 通常、ビジネスとノードの目的を説明するために使用します。
- Parameter: タスクスケジューリングの際、コードの変数に値を割り当てるために使用します。

たとえば、変数 pt=\${datetime} を使ってコード内で時間を指定する場合、ここで変数 に値を割り当てます。割り当てる値には、スケジューリング組み込み時間パラメーター " datetime=\$bizdate" を使用できます。

· Resource Group: ノードを実行するためのリソースグループを指定します。

各種ノードタイプのパラメーター値の割り当て形式

- ODPS SQL、ODPSPL、ODPS MR、XLIB タイプ: Variable name 1=Parameter 1
 Variable name 2=Parameter 2... 複数のパラメーターはスペースで区切ります。
- SHELL タイプ: Parameter 1 Parameter 2... 複数のパラメーターはスペースで区切ります。

頻繁に使用される時間パラメーターのいくつかは、組み込みスケジューリングパラメーターとし て提供されています。 パラメーターの詳細については、「パラメーターの設定」をご参照くださ い。

1.9.2 マニュアルノードパラメーターの設定

自動的にスケジュールされた時間にタスクが実行される際、動的に環境の変化に適用できるよう にするために、DataWorks ではパラメーター設定機能を提供します。 パラメーターを設定する 前に次の問題点について注意してください。 ・パラメーター内の "="の両端にはスペースを追加することができません。 正しい: bizdate=

\$bizdate

Basics ⑦					
	Node Name:	insert_data	Node ID:		
	Node Type:	ODPS SQL	Owner:	1.9	
	Description:				
	Parameters:	bizdete=\$datetime			0

・ 複数のパラメーター (存在する場合) は、スペースで区切ります。

Basics ⑦					
	Node Name:	insert_data	Node ID:		
	Node Type:	ODPS SQL	Owner:	1 11	
	Description:	Add spaces between the	e two para	meters.	
	Parameters:	bizdate=\$bizdate datetime=\$(yyyymmdd)			0

システムパラメーター

DataWorks では、2つのシステムパラメーターを提供し、次のとおり定義されます。

- ・ \${bdp.system.cyctime}: インスタンスのスケジュール実行時間を定義します。 デフォルト
 形式: yyyymmddhh24miss
- ・ \${bdp.system.bizdate}: インスタンスが計算される営業日を定義します。デフォルトの営業日は、実行日の前日です。デフォルトの表示形式は、yyyymmddです。

定義に応じて、ランタイムと営業日の計算式は次のとおりです。Runtime = Business date - 1

システムパラメーターを使用するためには、編集ボックスでシステムパラメーターを設定せず にコード内の **\${bizdate}** を直接参照します。システムは自動的にコード内のシステムパラメー ターの参照フィールドを置換します。

🗎 注:

周期タスクのスケジューリング属性は、スケジュールランタイムを使って設定されます。 した がって、インスタンスのスケジュールランタイムをバックトラックしたり、インスタンスのシス テムパラメーター値を取得したりできます。

例

ODPS_SQL タスクを毎日 00:00 から 23:59 間の毎時間実行するように設定します。 コードでシ ステムパラメーターを使用するには、次の文を実行します。

```
insert overwrite table tb1 partition(ds ='20180606') select
c1,c2,c3
from (
select * from tb2
where ds ='${bizdate}');
```

非 Shell ノードのスケジューリングパラメーターの設定

🧾 注:

SQL コードの変数名には、a-z、A-Z、数字、アンダースコアを使用できます。 変数名が date の場合、値 \$bizdate は自動的にこの変数に割り当てられます。スケジューリングパラメーター の設定で値を割り当てる必要はありません。 もし他の値が割り当てられていても、デフォルト で値 \$bizdate が自動的に割り当てられるため、コードではこの値は使用されません。

非 Shell ノードでは、\${variable name} をコードに追加します (この関数が参照されているこ とを示します)。 特定の値を入力し、値をスケジューリングパラメーターへ割り当てます。

たとえば、ODPS SQL ノードに対しては、**\${variable name}** をコードに追加します。ノードの パラメーター項目 "変数名=ビルトインスケジューリングパラメーター" を設定します。

コード内で参照されているパラメーターでは、スケジューリング中の解析値を追加する必要があ ります。



Shell ノードのスケジューリングパラメーターの設定

Shell ノードのパラメーター設定手順は、非 Shell ノードのパラメーター設定手順と似ていますが、規則が異なります。 Shell ノードでは、変数名はカスタマイズできず、**\$1,\$2,\$3...** という名前である必要があります。

たとえば、Shell ノードの場合、コード内の Shell 構文宣言は \$1、スケジューリングのノードパ ラメーター設定は、\$xxx (ビルトインスケジューリングパラメーター) です。 つまり、\$xxx の 値はコードの \$1 を置換するために使用されます。

コード内で参照されているパラメーターでは、スケジューリング中の解析値を追加する必要があ ります。





Shell ノードでは、パラメーター数が 10 に到達する場合、\${10} を使って変数を宣言する必要 があります。

変数値は、固定値です。

SQL ノードを例に説明します。 コードの **\${variable name}** では、ノードのパラメーター項目 "変数名="fixed value"" に設定します。

 $\exists - F$: select xxxxx type=' \${type}'

スケジューリング変数に割り当てられる値: type="aaa"

スケジューリング中では、コードの変数は、type='aaa'によって置換されます。

変数値は、ビルトインスケジューリングパラメーターです。

SQL ノードを例に説明します。 コードの **\${variable name}** では、パラメーター項目 変数名 = ノードのスケジューリングパラメーターとして設定します。

$\exists - F$: select xxxxx dt=\${datetime}

スケジューリング変数に割り当てられる値: datetime=\$bizdate

スケジューリング中、今日が 2017 年 7 月 22 日の場合、コードの変数は、dt=20170721 に置換されます。

ビルトインスケジューリングパラメーターの一覧

\$bizdate: 形式 yyyymmdd の営業日 注記: このパラメーターは広く使用され、ルーチンスケ ジューリングでは、デフォルトで前日となります。 たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$bizdate です。 今日が 2017 年 7 月 22 日とします。 ノードが本日実行され る場合、\$bizdate は pt=20170721 に置換されます。

たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$gmtdate です。 今日が 2017 年 7 月 22 日とします。 ノードが本日実行され る場合は、\$gmtdate は pt=20170722 に置換されます。

たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$gmtdate です。 今日が 2017 年 7 月 1 日とします。 ノードが本日実行され る場合は、\$bizdate は pt=20130630 に置換されます。

たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定 では、datetime=\$gmtdate です。 今日が 2017 年 7 月 1 日とします。 ノードが本日実行され る場合は、\$gmtdate は pt=20170701 に置換されます。

\$cyctime: タスクのスケジュールされた時間です。毎日のタスクでスケジュール時間が設定され ていない場合、**cyctime** は現在の日の **00:00** です。時間は、時間、分、秒に対して正確です。 通常は時間単位、分単位のスケジューリングタスクに使用されます。例: **cyctime=\$cyctime**

注:

\$[]と\${}を使って設定された時間パラメーターの違いについてご注意ください。\$bizdate: デフォルトでは、現在の日の前日の営業日です。\$cyctime: タスクのスケジュールされた時間 です。毎日のタスクでスケジュール時間が設定されていない場合、タスクは現在の日の 00:00 に実行されます。時間は、時間、分、秒に対して正確です。通常は時間単位、分単位のスケ ジューリングタスクに使用されます。たとえば、タスクが今日の 00:30 に実行されるようにス ケジュールされている場合、スケジュール時間は、yyyy-mm-dd 00:30:00 です。[]を使っ て時間パラメーターが設定されている場合、cyctime は実行のベンチマークとして使用されま す。使用に関する詳細は、次の手順をご参照ください。時間の算出方法は Oracle の算出方法 と同じです。データ作成時、置換後のパラメーター値は、営業日+1日です。たとえば、日付 が 20140510 を営業日として選択している場合、cyctime は 20140511 に置換されます。

\$jobid: タスクが属するワークフローの ID です。 例: jobid=\$jobid

\$nodeid: ノードの ID です。 例: nodeid=\$nodeid

\$taskid: タスクの ID です。つまりノードインスタンスの ID です。 例: taskid=\$taskid

\$bizmonth:形式 yyyymm の営業月です。

- ・ 営業月が現在の月と同じ場合、\$bizmonth = 営業月 1 です。そうでない場合は、\$bizmonth = 営業日の月です。
- たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター
 設定では、datetime=\$bizmonth です。 今日が 2017 年 7 月 22 日とします。 ノードが今
 日実行される場合、\$bizmonth は pt=201706 に置換されます。

\$gmtdate: 形式 **yyyymmdd** の現在の日です。 このパラメーターの値は、デフォルトでは今日 です。 データ作成時、入力した **gmtdate** は営業日 プラス **1** です。

カスタムパラメーター \${…}パラメーターの説明:

- ・ \$bizdate に基づいてカスタマイズされた時間形式では、yyyy は 4 桁の年、yy は 2 桁の月、 mm は月、dd は日を示します。パラメーターは意図したとおりに組み合わせることができま す。たとえば、\${yyyy}、\${yyyymm}、\${yyyymmdd}、\${yyyy-mm-dd}です。
- * \$bizdate は年、月、日に対して正確です。したがって、カスタムパラメーター \${……}は、
 年、月、日のみを表現できます。
- ・ 特定の期間のプラス、マイナスの期間を取得する方法:

次の N 年: \${yyyy+N}

前のN年: \${yyyy-N}

次のN月: \${yyyymm+N}

前のN月: \${yyyymm-N}

次のN週間: \${yyyymmdd+7*N}

前のN週間: \${yyyymmdd-7*N}

次のN日: \${yyyymmdd+N}

前のN日: \${yyyymmdd-N}

\${yyyymmdd}:形式 yyyymmdd の営業日です。 **\$bizdate** の値と一致します。

- 注記: \$bizdate の値と一致します。このパラメーターは広く使用されます。ルーチンスケジューリングではデフォルトで、前日の日付となります。このパラメーターの形式をカスタマイズすることができます。たとえば、\${yyyy-mm-dd}の形式を yyyy-mm-dd にします。
- たとえば、ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター 設定は、datetime=\${yyyymmdd} です。 今日が 2013 年 7 月 22 日とします。 ノードが今 日実行される場合、\${yyyymmdd} は pt=20130721 に置換されます。

\${yyyymmdd-/+N}: yyyymmdd プラス、またはマイナス N 日

\${yyyymm-/+N}: yyyymm プラス、またはマイナス N 月

\${yyyy-/+N}: 年 (yyyy) プラス、またはマイナス N 年

\${yy-/+N}:年(yy) プラス、またはマイナスN年

注記: yyyymmdd は営業日を示し、yyyy-mm-dd のように区切りをサポートします。 上記の パラメーターは、営業日の年、月、日に由来しています。

例:

- ODPS SQL ノードのコードでは、pt=\${datetime}です。 ノードのパラメーター設定では、 datetime=\${yyyy-mm-dd}です。 今日が 2018 年 7 月 22 日とします。 ノードが今日実行 される場合、\${yyyy-mm-dd}は pt=2018-07-21 に置換されます。
- ODPS SQL ノードのコードでは、pt=\${datetime} です。 ノードのパラメーター設定では、 datetime=\${yyyymmdd-2} です。 今日が 2018 年 7 月 22 日とします。 ノードが今日実行 される場合、\${yyyymmdd-2} は pt=20180719 に置換されます。
- ODPS SQL ノードのコードでは、pt=\${datetime}です。 ノードのパラメーター設定では、 datetime=\${yyyymm-2}です。 今日が 2018 年 7 月 22 日とします。 ノードが今日実行さ れる場合、\${yyyymm-2} は pt=201805 に置換されます。
- ODPS SQL ノードのコードでは、pt=\${datetime}です。 ノードのパラメーター設定では、 datetime=\${yyyy-2}です。 今日が 2018 年 7 月 22 日とします。 ノードが今日実行される 場合、\${yyyy-2} は pt=2018 に置換されます。

ODPS SQL ノード設定では、複数のパラメーターに値が割り当てられます。たとえば、 startdatetime=\$bizdate enddatetime=\${yyyymmdd+1} starttime=\${yyyy-mm-dd} endtime=\${yyyy-mm-dd+1} です。

例: (\$cyctime=20140515103000 と想定します)

\$[yyyy] = 2014, \$[yy] = 14, \$[mm] = 05, \$[dd] = 15, \$[yyyy-mm-dd] = 2014-05-15, \$[hh24:mi:ss] = 10:30:00, \$[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00

```
• $[hh24:mi:ss - 1/24] = 09:30:00
```

- \$[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
- \$[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-1509:30:00
- \$[add_months(yyyymmdd,-1)] = 2014-04-15
- \$[add_months(yyyymmdd,-12*1)] = 2013-05-15

```
• $[hh24] =10
```

• \$[mi] =30

```
パラメーター $cyctime をテストする方法
```

インスタンスが実行された後、ノードを右クリックしノードの属性を確認します。 スケジュール された時間がインスタンスが周期的に実行される時間かどうかを確認します。

パラメーター値がスケジュールした時間マイナス1時間に置換された後の結果です。

1.10 コンポーネントの管理

1.10.1 コンポーネントの使用

開発効率を向上させるため、データタスク開発者はプロジェクトとテナントから提供されるコン ポーネントを使って、データプロセスノードを作成します。

- ・ ローカルプロジェクトのメンバーが作成したコンポーネントは [Project Components] の下 にあります。
- ・テナントメンバーが作成したコンポーネントは [Public Components] の下にあります。

コンポーネントの使用方法については、「SQL コンポーネントノード」をご参照ください。

インターフェイスの機能

1 2 3	SQL component model author:wangdan	X Basics		
4 5 6	create time:2018-09-03 00:11:21 document: <u>https://help.aliyun.com/document_detail/30290.html</u> 	Component Name :	myComponent	
7 8 9 10	<pre>insert overwrite table @@{my_output_table} partition (ds-'\${bizdate}') select</pre>	* Owner : Description :	wangdan	
11 12 13	from @@(mv input table)	Input Parameters (?)		
14 15	<pre>where category in ('@@(my_input_parameter1)', '@@(my_input_param AND substr(pt, 1, 8) in ('\$(bizdate)')</pre>	* Parameter Name :	* Type :	String ~
16 17		Description :		
		Default Value :		
		Output Parameters(?)		
	$\overline{\mathbf{x}}$	* Parameter Name :	* Type :	String 🗸
	5.7 KV	Description :		
		Default Value :		

インターフェイスの機能は次のとおりです。

No.	機能	説明
1	Save	クリックして、現在のコンポーネントの設定を保存しま す。

No.	機能	説明
2	Steallock Edit	現在のコンポーネントの所有者ではない場合、クリック し、ノードを steallock 編集します。
3	Submit	クリックして、現在のコンポーネントを開発環境へ送信 します。
4	Publish Component	クリックして、ユニバーサルグローバルコンポーネント をテナント全体へ発行します。テナントのすべてのユー ザーがパブリックコンポーネントを表示し使用すること ができます。
5	Resolve Input and Output Parameters	クリックして、現在のコードの入出力パラメーターを解 決します。
6	Pre-compile	クリックして、現在のコンポーネントのカスタムおよび コンポーネントパラメーターを編集します。
7	Run	クリックして、開発環境でコンポーネントをローカルで 実行します。
8	Stop Run	クリックして、実行中のコンポーネントを停止します。
9	Format	クリックして、現在のコンポーネントをキーワード別に ソートします。
10	Parameter settings	クリックして、コンポーネント情報、入力パラメーター の設定、出力パラメーターの設定を表示します。
11	Version	クリックして、現在のコンポーネントの送信および解放 履歴を表示します。
12	Reference Records	クリックして、コンポーネントの使用履歴を表示しま す。

1.10.2 コンポーネントの作成

コンポーネントの定義

コンポーネントは、複数の入出力パラメーターを含む SQL コードプロセステンプレートです。 SQL コードプロセスを処理するため、1 つ以上のソースデータテーブルがインポート、フィル タ、結合、および集約され、新しいビジネスで必要とされるターゲットテーブルを形成します。

コンポーネントの値

実際のビジネスでは、SQL コードプロセスは類似しています。 処理上にある入力と出力テーブル は、同じまたは互換性のある構造になっていますが、名前が異なります。 この場合、コンポーネ ントの開発者は、このような SQL プロセスを SQL コンポーネントノードへ、SQL プロセスにあ る変数入出力テーブルを入出力パラメーターへ要約し、SQL コードを再利用します。 SQL コンポーネントノードを使用する際、コンポーネントユーザーはコンポーネントの一覧から ビジネスフローを選択するように、コンポーネントを選択し、これらコンポーネント向けのビジ ネスにある特定の入出力テーブルを設定します。コードのコピーを繰り返し行わずに新しい SQL コンポーネントノードを生成します。これは、開発効率を大幅に向上させ、繰り返し開発を行う ことを防ぐことができます。生成後のSQL コンポーネントノードの発行とスケジューリングは、 共通 SQL ノードの発行とスケジューリングと同じです。

コンポーネントの構成

関数の定義と同じように、コンポーネントは入力パラメーター、出力パラメーター、コンポーネ ントコードプロセスによって構成されています。

コンポーネント入力パラメーター

コンポーネント入力パラメーターには、名前、タイプ、説明、定義といった属性が含まれます。 パラメーターのタイプは、テーブルまたは文字列です。

- ・テーブルタイプのパラメーターは、コンポーネントプロセスで参照されるテーブルを指定します。コンポーネントを使用する際、コンポーネントユーザーは特定のビジネスで必要とされるテーブルにパラメーターを設定します。
- ・ 文字列タイプのパラメーターは、コンポーネントプロセスでの変数制御パラメーターを指定します。たとえば、特定のプロセスの結果テーブルが各リージョンのトップN都市の販売額のみを出力する場合、値Nは文字列パラメーターで指定することができます。

特定のプロセスの結果テーブルがその省の販売額の総額を出力する必要がある場合、省文字列 タイプのパラメーターを設定し、異なる省を指定し、その指定した省の販売額を取得します。

- パラメーターの説明は、コンポーネントプロセスのパラメーターの役割を指定します。
- パラメーターの定義は、テーブル構造のテキスト定義で、これはテーブルタイプのパラメー ターに対してのみ要求されます。この属性を指定する場合、コンポーネントプロセスが適 切に実行されるように、コンポーネントユーザーはテーブルパラメーターで定義されている フィールド名とタイプに互換性のある入力テーブルを提供する必要があります。そうでない 場合、入力テーブル内に指定されたフィールドが見つからないため、コンポーネントプロセス が実行される際にエラーが報告されます。入力テーブルには、テーブルパラメーターで定義 されたフィールド名とタイプが含まれている必要があります。フィールドとタイプの順序は 異なっていても、また入力テーブルに他のフィールドが含まれていてもかまいません。定義 は、参照目的のみです。ユーザーに対してガイダンスを提供し、すぐに強制的に確認する必 要はありません。
- ・ 推奨されるテーブルパラメーターの定義形式は次のとおりです。

Field 1 name Field 1 type Field 1 comment
Field 2 name Field 2 type Field 2 comment Field n name Field n type Field n comment

例:

area_id string 'Region ID' city_id string 'City ID' order_amt double 'Order amount'

コンポーネント出力パラメーター

- ・コンポーネント出力パラメーターには、名前、タイプ、説明、定義といった属性が含まれます。パラメータータイプは、テーブルのみです。文字列出力パラメーターは論理的な意味を 持ちません。
- ・テーブルタイプパラメーター:コンポーネントプロセスから生成されるテーブルを指定します。コンポーネントを使用する際、コンポーネントユーザーは、特定のビジネスに対してコンポーネントプロセスが生成する結果テーブルに対してパラメーターを設定します。
- ・パラメーターの説明:コンポーネントプロセスのパラメーターの役割を指定します。
- パラメーターの定義: テーブル構造のテキスト定義です。この属性を指定する場合、コンポーネントプロセスが適切に実行されるように、コンポーネントユーザーはテーブルパラメーターで定義されているフィールド名とタイプの数と同じ数を持つ出力テーブルとパラメーターを提供する必要があります。そうでない場合、フィールド数が一致しない、またはタイプに互換性がないため、コンポーネントプロセスが実行される際エラーが報告されます。出力テーブルのフィールド名は、テーブルパラメーターで定義された名前とは一致しません。定義は参照目的のみです。ユーザーに対してガイダンスを提供し、すぐに強制的に確認する必要はありません。
- ・ 推奨されるテーブルパラメーターの定義形式は次のとおりです。

Field 1 name Field 1 type Field 1 comment Field 2 name Field 2 type Field 2 comment Field n name Field n type Field n comment

例:

area_id string 'Region ID' city_id string 'City ID' order_amt double 'Order amount' rank bigint 'Rank'

コンポーネントプロセス体

プロセス体のパラメーターの参照形式は、 @@{parameter name}です。

要約 SQL ワーキングプロセスのコンパイルによって、プロセス体は入力パラメーターに基づい て指定された入力テーブルを制御し、ビジネス値を持つ出力テーブルを生成します。 コンポーネントプロセスの開発には特定のスキルが必要となります。入出力パラメーターの異な る値が正しく実行可能な SQL コードを生成できるように、入出力パラメーターはコンポーネン トプロセスコードで良く使用されます。

コンポーネントの作成例

次の図に示すとおり、コンポーネントを作成します。

Components	C C	C myComponent x C testSQLComponent x D ftp_sync x Business Flow x Solution	X Sq rpt_user_ini
Project-specific		" A F 🖶 🗃 🖸 💿 📁 🗱	
A myComponent wangdar		<pre>1 SQL component model 2</pre>	×
			取制

ソーステーブルスキーマの定義

営業データのソース MySQL スキーマの定義は次の表で説明するとおりです。

フィールド名	フィールドタイプ	フィールドの説明
order_id	varchar	注文 ID
report_date	datetime	注文日
customer_name	varchar	顧客名
order_level	varchar	注文グレード
order_number	double	注文数
order_amt	double	注文金額
back_point	double	割引
shipping_type	varchar	輸送モード
profit_amt	double	利益額
price	double	単価
shipping_cost	double	輸送コスト
area	varchar	リージョン

フィールド名	フィールドタイプ	フィールドの説明
province	varchar	省
city	varchar	市
product_type	varchar	プロダクトタイプ
product_sub_type	varchar	プロダクトサブタイプ
product_name	varchar	プロダクト名
product_box	varchar	プロダクト梱包箱
shipping_date	Datetime	輸送日

コンポーネントのビジネス的な意味合い

コンポーネント名: get_top_n

コンポーネントの説明:

コンポーネントプロセスでは、指定した営業データテーブルを入力パラメーター (テーブルタイ プ) として使用し、トップ都市の数を入力パラメーター (文字列タイプ) として使用します。都市 は販売金額別にランク付けされます。 このように、コンポーネントユーザーは各リージョンの指 定したトップ N 都市のランキングを簡単に取得することができます。

コンポーネントパラメーターの定義

入力パラメーター1:

パラメーター名: myinputtable type: table

入力パラメーター2

パラメーター名: topn type: string

入力パラメーター3

パラメーター名: myoutput type: table

パラメーターの定義

area_id string

city_id string

order_amt double

rank bigint

テーブル作成文:

CREATE TABLE IF NOT EXISTS company_sales_top_n (area STRING COMMENT 'Region', city STRING COMMENT 'City', sales_amount DOUBLE COMMENT 'Sales amount', rank BIGINT COMMENT 'Rank') COMMENT 'Company sales ranking' PARTITIONED BY (pt STRING COMMENT '') LIFECYCLE 365;

コンポーネントプロセス体の定義

```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
    SELECT r3.area_id,
    r3.city_id,
    r3.order_amt,
    r3.rank
from (
SELECT
    area_id,
    city_id,
    rank,
    order_amt_1505468133993_sum as order_amt ,
    order_number_1505468133991_sum,
    profit_amt_1505468134000_sum
FROM
    (SELECT
    area_id,
    city_id,
    ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_
1505468133993_sum DESC)
AS rank,
    order_amt_1505468133993_sum,
    order_number_1505468133991_sum,
    profit_amt_1505468134000_sum
FROM
    (SELECT area AS area_id,
     city AS city_id,
     SUM(order_amt) AS order_amt_1505468133993_sum,
     SUM(order_number) AS order_number_1505468133991_sum,
     SUM(profit_amt) AS profit_amt_1505468134000_sum
FROM
    @@{myinputtable}
WHERE
    SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
GROUP BY
    area,
    city )
    r1 ) r2
WHERE
    r2.rank >= 1 AND r2.rank <= @@{topn}
ORDER BY
    area_id,
    rank limit 10000) r3;
```

コンポーネントの共有スコープ

共有スコープは、プロジェクトコンポーネントとパブリックコンポーネントの2つがあります。

コンポーネントを発行した後、デフォルトでプロジェクト内に表示されます。 コンポーネント開 発者は、[Publish Component] アイコンをクリックし、ユニバーサルグローバルコンポーネン トを全体のテナントへ発行します。テナント内のすべてのユーザーがパブリックコンポーネント を表示し使用することができます。 コンポーネントがパブリックかどうかは、次の図のアイコン が表示されているかどうかによります。

1 2 3	SQL component model author:wangdan	X Basics				
4 5 6 7	<pre>create time:2018-09-03 00:11:21document: <u>https://help.aliyun.com/document_detail/30200.html</u></pre>	* Component Name : * Owner :	myComponent wangdan			
8 9 10	<pre>insert overwrite table @@{my_output_table} partition (ds-'\${bizdate}') select </pre>	Description :				
12	from Aler input table)	Input Parameters(7)				
14 15	<pre>where category in ('@@(my_input_parameter1)', '@@(my_input_param AND substr(pt, 1, 8) in ('\$(bizdate)')</pre>	* Parameter Name :		• Type :	String	
16 17		Description :				
		Default Value :				
		Output Parameters(?)				
	$\overline{\mathbf{x}}$	* Parameter Name :		• Type :	String	
	5.7 2.9	Description :				
		Default Value :				

コンポーネントの使用

開発されたコンポーネントはどのように使用しますか。 詳細については、次をご参照ください。 コンポーネントの使用

コンポーネントの参照履歴

コンポーネント開発者は、[Reference Records] タブをクリックし、コンポーネントの参照履 歴を表示します。

Proje ct Na me	N o d ID	Nod e me	Referenced Co mponent Name	O w n e r	Cre ate d A t	Developm ent Versio n	Producti on Versi on	ameters Ve
			No data	8				ersion
								Ref
< 1	>							ference Records

1.11 クエリ

一時的なクエリは、コード編集の使用を容易にし、ローカルコードの実際の状態が想定通りに なっているかどうかをテストし、コードステータスを確認します。 したがって、一時的なクエリ は送信、解放、スケジューリングパラメーターの設定をサポートしません。 スケジューリングパ ラメーターを使用するには、[Data development] または [Manual business flow] でノード を作成します。

フォルダの作成

1. 左側のナビゲーションウィンドウで、[Queries] をクリックし、[folder] を選択します。



2. フォルダ名を入力し、フォルダディレクトリを選択します。[Submit] をクリックします。

Create Folder			×
Folder Name :	testdoc		
Destination Folder :	Queries	*	
		Submit	Cancel

注:

マルチレベルのフォルダディレクトリがサポートされています。 したがって、作成した別の フォルダにフォルダを格納することができます。

ノードの作成

一時的なクエリは、SHELL および SQL ノードのみをサポートします。

■ Querles 烇鼠口CO ≦ myC	Component
Enter a file or creator name	€ [
★ v Queries	SQL
Sq select_01 Me锁定 08-31 16:12 3	auth
Sq test birdbd锁定 08-29 18:21 4 ↓ testdoc	crea docu
CreateCreateNode ID > ODPS	S SQL
Create Folder Shell	
# Rename 9	partit
Delete 11	serect

ODPS SQL ノードを例に説明します。フォルダ名を右クリックし、**[Create Node]** > **[ODPS SQL]** を選択します。

Sq test	sqL O
Ľ	
	odps sql
	author:wangdan
	create time:2018-09-03 12:55:16
	show TABLES;

No.	機能	説明	
1	Save	クリックして、入力したコードを保存します。	
2	Steallock Edit	ノードの所有者ではないユーザーがクリックして、ノー ドを編集します。	
3	Run クリックしてコードをローカルで (開発環境内で) 実行ます。		
4	Advanced Run (with Parameters)	クリックして、コードに設定されたパラメーターを使っ て、現在のノードのコードを実行します。	
		注 注: Advanced Run は Shell ノードでは利用できません。	
5	Stop Run	クリックして、実行中のコードを停止します。	
6	Reload	クリックして、ページを更新し、リロードし、最後に保 存された状態を復元します。 保存されていないコンテン ツは失われます。	
		注: 設定センターでキャッシュを有効にしている場合、 ページを更新した後、保存されていないコードが キャッシュされたことを示すメッセージが表示されま す。必要なバージョンを選択します。	
7	Format	クリックして、現在のノードコードをキーワード形式で ソートします。 コード行が長すぎるときに使用します。	

1.12 ランニングログ

ランニングログ (Running Log) ページは、過去3日間にローカルで実行されたすべてのタスクの履歴を表示します。 クリックして、タスクの履歴を表示し、タスクステータス別に実行履歴をフィルタします。



ランニングログは、3日分のみ取得できます。

ランニングログの表示

1. クリックして、[Running Log] ページへ移動します (デフォルトでは、すべてのステータスのタスクが表示されます)。



2. ドロップダウンリストボックスをクリックし、タスクフィルタ条件を選択します。

	Runtime Log C
	All
*	All
R	Succeeded
B	Failed
-	Waiting for Resources
	Pending for Running
Ħ	Running
2	Stopped
∱×	⊘ 08-31 10:25:37 select * from rpt_user_in
Ħ	⊘ 08-31 10:15:44 –odps sql _***********************
	⊘ 08-31 10:12:47 –odps sql –************************************
	⊙ 08-31 10:06:15 CREATE TABLE IF NOT

3. ターゲットランニング履歴をクリックします。 ランニングログページでは、ランニング履歴の ログが表示されます。

ログを一時的ファイルへ保存します。

ランニング履歴の SQL 文を保存するには、[Save] アイコンをクリックして、一時的ファイルに 実行された SQL 文を保存します。

ファイル名とディレクトリを入力し、[Submit] をクリックします。

1.13 **パブリックテーブル** (Public Table)

[Public Table] エリアでは、現在のテナントにあるすべてのプロジェクトで作成されたテーブル を表示できます。



- Project: プロジェクト名 プレフィックス "odps." が、各プロジェクト名に追加されています。
 たとえば、プロジェクト名が test の場合、"odps.test" と表示されます。
- ・ Table Name: プロジェクトのテーブル名です。

テーブル名をクリックして、テーブルの列やパーティション情報を表示し、テーブルデータをプ レビューします。

- Column Information: クリックして、テーブルのフィールド数、フィールドタイプ、フィー ルドの説明を表示します。
- Partition Information: クリックして、テーブルのパーティション情報とパーティション数 を表示します。最大 6,000 個のパーティションが許可されています。 ライフサイクルを設定 した場合、パーティションの実際の数は、ライフサイクルに依存します。
- ・ Data Preview: クリックして、現在のテーブルのデータをプレビューします。

環境切り替え

Table Management 同様に、Public Table は開発環境と運用環境をサポートしています。 現 在の環境は青で表示されます。 照会する環境をクリックすると、対応する環境が表示されます。



1.14 テーブルの管理

テーブルの作成

1. ページの左上隅にある [Table Management] をクリックします。

2. [+] アイコンを選択し、テーブルを作成します。

	Tables 📑 (C
	Enter a table name or description Create	í
*	✓ ■ Tables	
R	> 📄 one_level	
B	> Dthers	
۲		
Ħ		
5		
f×	Production	
đ	Environment	

3. テーブル名を入力します。MaxCompute テーブルのみが現在サポートされていま す。[Submit] をクリックします。

yeste ;	Create Table			×
	Database Type :	odps		
	Table Name :	test_table1		
			Submit	Cancel

- 4. 基本的な属性を設定します。
 - · Chinese Name: 作成するテーブルの中国語名です。
 - ・ Level-1 Topic: 作成するテーブルの level-1 ターゲットフォルダの名前です。
 - ・ Level-2 Topic: 作成するテーブルの level-2 ターゲットフォルダの名前です。
 - · Description: 作成するテーブルの説明です。
 - [Create Topic] をクリックします。表示された [Topic Management] ページ
 で、level-1 と level-2 トピックを作成します。



5. DDL モードでテーブルを作成します。

[DDL Mode] をクリックします。表示されたダイアログボックスで、標準テーブル作成文を 入力します。

テーブル作成 SQL 文を編集した後、[Generate Table Structure] をクリックします。 [Basic Attributes]、[Physical Model Design]、[Table Structure Design] エリアの情 報は自動的に入力されます。 6. GUI でテーブルを作成します。

DDL モードでのテーブルの作成が適切ではない場合、GUI で次の設定を行ってテーブルを作成します。

- Physical model design
 - Table type: [Partitioned Table] または [Non-partitioned Table] を設定します。
 - Life Cycle: MaxCompute のライフサイクル機能です。Life Cycle で指定した周期 (単位:日)でアップデートされていないテーブル(またはパーティション)のデータはク リアされます。
 - Level: [DW]、[ODS]、または [RPT] を設定します。
 - Physical Category: [Basic Business Layer]、[Advanced Business Layer]、または [Other] を設定します。 [Create Level] をクリックします。表示された [Level Management] ページで、レベルを作成します。
- Table structure design
 - English Field Name: フィールドの英語名で、文字、数字、アンダースコア (_) を含めることができます。
 - Chinese Name: フィールドの中国語の略名です。
 - Field Type: MaxCompute データタイプで、String、Bigint、Double、Datetime、Boolean 型のみです。
 - Description: フィールドの詳細説明です。
 - Primary Key: 選択し、フィールドが主キーか、結合主キーにあるフィールドかを示し ます。
 - [Add Field] をクリックし、新しいフィールドに列を追加します。
 - **[Delete Field]** をクリックし、作成したフィールドを削除します。

注:

作成したテーブルからフィールドを削除し、テーブルを再度送信した場合、現在のテー ブルを破棄して、同じ名前のテーブルを新しく作成する必要があります。 この操作は 運用環境では行えません。

- [Move Up] をクリックし、作成したテーブルのフィールドの順番を調整します。 しか し、作成したテーブルのフィールドの順番を調整するには、現在のテーブルを破棄し て、同じ名前のテーブルを新しく作成する必要があります。 この操作は運用環境では行 えません。

- [Move Down] をクリックします。操作は [Move Up] と同じです。
- [Add Partition] をクリックし、現在のテーブルにパーティションを作成します。作成したテーブルにパーティションを追加するためには、現在のテーブルを破棄して、同じ名前のテーブルを新しく作成する必要があります。この操作は運用環境では行えません。
- [Delete Partition] をクリックしてパーティションを削除します。 現在のテーブルから パーティションを削除するには、現在のテーブルを破棄して、同じ名前のテーブルを新 しく作成する必要があります。 この操作は運用環境では行えません。
- Action:新しいフィールドの送信、フィールドの削除、また他の属性を編集します。

プロパティには、主にシステムに対して検証ロジックを生成するために提供されるデー タ品質に関連する情報が含まれています。 データプロファイル、SQL スキャン、テスト ルール作成などのシナリオで使用されます。

- 0 Allowed: 選択すると、フィールド値をゼロにすることができます。 このオプションは、bigint と Double 型フィールドのみに適用されます。
- Negative Value Allowed: 選択すると、フィールド値を負の値にすることができま す。 このオプションは、bigint とDouble 型フィールドのみに適用されます。
- Security Level: セキュリティレベルは0-4 です。数字が大きくなると、セキュリ ティ要件も高くなります。セキュリティレベルがデジタル要件を満たさない場合、 フォームの対応するフィールドへアクセスできません。
- Unit: 金額です。ドル、またはセントにすることができます。 このオプションは金額 に関連しないフィールドには不要です。
- Lookup Table Name/Kay Value: メンバータイプやステータスといった列挙型 フィールドに適用されます。 このフィールドに対応した辞書テーブル (またはディメ ンションテーブル) の名前を入力します。 たとえば、メンバーステータスに対応した 辞書テーブル名は、"dim_user_status" です。 グローバルユニークな辞書テーブル

を使用する場合、辞書テーブル内のフィールドに対応する key_type を入力します。 たとえば、メンバーステータスの対応キー値は、"AOBAO_USER_STATUS" です。

- Value Range: 現在のフィールドの最大値と最小値です。 Bigint と Double の フィールドにのみに適用されます。
- Regular Expression Verification: 現在のフィールドで使用する正規表現です。 たとえば、フィールドが携帯電話番号で、正規表現 (またはさらに厳格な制限) で 11 桁の数字に制限することができます。
- Maximum Length: フィールド値の最大文字数です。 String 型のフィールドにの み適用されます。
- Date Precision: 日付の精度です。[Hour]、[Day]、[Month]、[others]のいずれかに設定できます。たとえば、フィールド値が 2014-08-01であっても (精度は Dayのように見えます)、月次サマリーテーブル "month_id" の精度は Month です。 Datetime または String タイプの日付値に適用されます。
- Date Format: String タイプの日付値にのみ適用されます。 フィールドに実際に格 納される日付値の形式は、yyyy-mm-dd hh:mm:ss のようになります。
- KV Primary Separator/Secondary Separator: キー値ペアが組み合わされた大 規模なフィールド (String型) に適用されます。たとえば、プロダクト拡張属性に " key1:value1;key2:value2;key3:value3;..."といった値がある場合、セミコロン (;) はフィールドの主となるセパレーターで、キー値ペアを区切ります。コロン (:) は 二次的なセパレーターで、キー値ペアのキーと値を区切ります。
- Partition Field Design: このオプションは、[Physical Model Desing] エリアの [
 Partition Type] が[Partitioned Table] に設定されている場合にのみ表示されます。
- · Field Type: すべてのフィールドに対して、String タイプを使用することを推奨します。
- Date Partition Format: パーティションフィールドが日付 (データタイプは String) の場合、yyyymmmdd といった日付形式を選択または入力します。
- Date Partition Granularity: たとえば、[Day]、[Month]、[Hour] です。必要に応じて、パーティション細分性を設定します。デフォルトでは、複数のパーティション細分性が必要な場合、細分性が高いほど、パーティションのレベルが高くなります。たとえば、3つのパーティション(時、日、月)が存在する場合、複数のパーティションのリレーションシップは、level-1パーティション(月)、level-2パーティション(日)、level-3パーティション(時)です。

テーブルを送信します。

テーブル構造情報を編集した後、新規テーブルを開発環境と運用環境へ送信します。

- [Load from Development Environment] をクリックします。テーブルが開発環境へ送信 されると、ボタンはハイライトされます。ボタンをクリックすると、開発環境で作成された テーブルの情報は、現在のページの情報へ上書きされます。
- [Submit to Development Environment] をクリックします。システムは、現在の編集 ページで必要な項目がすべて設定されているかどうかを確認します。未入力の項目があった場 合、テーブルの送信を禁止するアラームが報告されます。
- ・ [Load from Production Environment] をクリックします。 運用環境へ送信したテーブル の詳細情報は、現在のページの情報へ上書きされます。
- [Create in Production Environment] をクリックします。テーブルは、運用環境のプロジェクトに作成されます。

タイプ別のテーブルの照会

[Table Management] ページでは、[Development Environment] または [Production Environment] のどちらかを選択してテーブルを照会します。 クエリ結果はフォルダーのト ピック別にソートされます。

- [Development Environment] を選択する場合、開発環境にあるテーブルのみを照会できます。
- ・ [Production Environment] を選択する場合、運用環境にあるテーブルを照会します。 運用 環境にあるテーブルを操作するときは注意してください。

1.15 外部テーブル

外部テーブルの概要

外部テーブルを使用する前に、次の概念について理解しておく必要があります。

名前	説明
Object Storage Service (OSS)	OSS は、[Standard]、[Infrequent Access]、[Archive] ストレー ジタイプをサポートします。データストレージとアクセスに必要な 要件が異なる場合のサービスシナリオに適用されます。さらに、 OSS は、Apache Hadoop、E-MapReduce、BatchCompute、 MaxCompute、Machine Learning Platform for AI (PAI)、Data Lake Anaytics、Function Compute、その他 Alibaba Cloud サー ビスに対するシームレスな統合をサポートします。

MaxCompute	ビッグデータコンピューティングサービスは、高速で、データ倉庫を すべて管理するソリューションです。 OSS と組み合わせて使用すると き、大規模なデータを低コストで効果的に分析し処理することができま す。その処理能力から、フォレスターは、 MaxCompute を世界有数 のクラウドベースデータ倉庫の1つとしています。
MaxCompute の外部テーブル	この機能は、MaxCompute v2.0 の計算フレームワークの新しい世 代に基づいています。これは、MaxCompute の内部テーブルヘデー タをロードすることなく、OSS に格納されているデータを直接照会し ます。データを移行する際、時間と労力のみでなく、重複データのス トレージコストも節約します。MaxCompute の外部テーブルを使っ て、同様の方法で Table Store に格納されているデータを照会しま す。

次の図では、外部テーブルの処理アーキテクチャを示します。



[OSS -> MaxCompute -> OSS] Data computing link

現在、MaxCompute では OSS や Table Store といった非構造データのストレージでの外部 テーブルの処理をサポートします。 データのフローと処理ルールに基づいて、非構造データ処理 フレームワークの主な機能がデータをインポートしエクスポートし、MaxCompute の入出力を 接続することを理解できます。 次の例では、OSS にある外部テーブルに適用される処理ルールを 説明します。

- OSS に格納されているデータは、非構造データ処理フレームワークを使って変換され、 InputStream Java クラスを使ったユーザー定義のインターフェイスへ送られます。 抽出 ルールを実装するには、入力ストリームを読み込み、解析、変換、計算する必要があります。 データはレコード形式に戻す必要があります。この形式は MaxCompute の一般的な形式で す。
- 2. これらのレコードは、MaxCompute に内蔵されている SQL エンジンに基づいて、新しいレ コードを生成するために構造データ処理に使用されます。
- 3. Output Stream Java クラスを使ってレコードデータが出力され、MaxCompute によって OSS ヘインポートされる前に更に計算を実行することができます。

MaxCompute によって提供される DataWorks を使って GUI で外部テーブルを作成、検索、 照会、設定、処理、および分析することができます。

ネットワークとアクセス権限の付与

MaxCompute は OSS から独立しているため、異なるクラスターでのネットワーク接続 は、OSS に格納されているデータへアクセスするための MaxCompute の能力に影響する場 合があります。 プライベートエンドポイント (-internal.aliyuncs.com で終わる) を使っ て、MaxCompute から OSS に格納されているデータへアクセスすることを推奨します。

OSS に格納されているデータへアクセスするには、MaxCompute への権限が必要となります。 MaxCompute は、Alibaba Cloud が提供する Resource Access Management (RAM) と Security Token Service (STS) を使ったセキュアなデータアクセスを保証します。 テーブル作 成者として、MaxCompute に対してSTS トークンを申請します。 したがって、MaxCompute と OSS は、同じ Alibaba Cloud アカウントで使用する必要があります。 同様の権限付与プロ セスは、Table Store に格納されているデータへアクセスする際にも適用されます。

1. STS の権限付与

MaxCompute が OSS に格納されているデータに対する直接的なアクセス権を必要とする 場合、まず RAM ユーザーに対し OSS アクセス権を付与する必要があります。Security Token Service (STS) は Alibaba Cloud が提供するセキュリティトークン管理サービスで す。Resource Access Management (RAM) に基づいたプロダクトです。権限付与された RAM ユーザーは、カスタム有効性と共にトークンを発行し、STS からアクセスすることが できます。 アプリケーションは、Alibaba Cloud API を直接呼び出し、リソースを操作する ことができます。詳細については、「OSS アクセスへの STS の権限の付与」をご参照くださ い。次の方法のいずれかを使ってアクセス権限を付与します。

MaxCompute と OSS が 同じ Alibaba Cloud アカウントである場合、ログインし、ワンクリック権限付与を行います。次の図に示すとおり、[Data Development] と [Create

Table] をクリックして、ワンクリック権限付与ページへ移動します。 詳細は、「」をご参照 ください。

Physical Model Partitioning :	O Partitioned Table 💿 Non- Partitioned Table	Time-to-Live :				
Table Level :	Select an option.	Table Category :	Select an option.	Create Level	C	
Table Type :	🔿 Internal Table 🔹 External Table					
Storage Space Address :					Select an option.	Authorize
Note: If you need to modify role perm	issions, please go to the RAM Console. Role Ma	nagement. If you do not config	ure it correctly, the following ro	le: ODPS will not be able to obt	ain the required permissions.	×
ODPS needs your permission Authorize ODPS to use the following	on to access your cloud resources, roles to access your cloud resources,					
AliyunODPSDefaultRole Description: ODPS默认使用此角的 Permission Description:	色来访问您在其他云产品中的资源					×

 カスタム権限の付与まず、RAM を使って OSS ヘアクセスする権限を MaxCompute へ 付与します。RAM コンソールヘログインします (MaxCompute と OSS が違う Alibaba Cloud アカウントである場合、そのアカウントを使って OSS ヘログインします)。 [Role Management] ページへ移動し、[Create Role] をクリックします。 [Role Name] の値 を「AliyunODPSDefaultRole」または「AliyunODPSRoleForOtherUser」に設定しま す。

[Role Details] を設定します。

[Role Authorization Policies] を設定します。OSS アクセス権を付与するために必要な「AliyunODPSRolePolicy」ポリシーを検索します。「AliyunODPSRolePolicy」 ポリシーをロールに添付します。[Search and Attach] を使ってこのポリシーを探 すことができない場合、[Input and Attach] を使ってロールに権限を付与します。 「AliyunODPSRolePolicy」ポリシーのポリシーコンテントは次のとおりです。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:ListBuckets",
        "oss:GetObject"
        "oss:ListObjects"
        "oss:PutObject",
        "oss:DeleteObject"
        "oss:AbortMultipartUpload",
        "oss:ListParts"
        」,
        "Resource": "*"
        "Effect": "Allow"
  },
      "Action": [
        "ots:ListTable".
        "ots:DescribeTable",
        "ots:GetRow",
        "ots:PutRow"
        "ots:UpdateRow",
        "ots:DeleteRow"
        "ots:GetRange",
        "ots:BatchGetRow"
        "ots:BatchWriteRow"
        "ots:ComputeSplitPointsBySize"
      ],
"Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

2. Data Integration での OSS データソースの使用

Data Integration ですでに作成されている OSS データソースを直接使用することができま

す。

外部テーブルの作成

1. DDL 文を使ってテーブルを作成します。

[Data Development] ページへ移動します。「テーブルの管理」を参照し、DDL 文を使って テーブルを作成します。 ODPS 構文に従う必要があります (「テーブルの操作」 をご参照く ださい)。 STS 権限が付与されている場合、odps.properties.rolearn 属性を含める必要 はありません。 次の例では、DDL 文を使ってテーブルを作成する方法を説明します。 文の EXTERNAL キーワードは、このテーブルが外部テーブルであることを示しています。

CREATE EXTERNAL TABLE IF NOT EXISTS ambulance data csv external(vehicleId int, recordId int, patientId int, calls int, locationLatitute double, locationLongtitue double, recordTime string, direction string STORED BY 'com.aliyun.odps.udf.example.text.TextStorageHandler' --The STORED BY clause specifies the StorageHandler for the correspond ing file format. This clause is required. with SERDEPROPERTIES ('delimiter'='\\|', --The SERDEPROPERITES clause specifies the parameters used when serializing or deserializing data. These parameters are passed into the code of Extractor through DataAttrib utes. This clause is optional. 'odps.properties.rolearn'='acs:ram::xxxxxxxxxxx:role/aliyunodps defaultrole' LOCATION 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/ Demo/SampleData/CustomTxt/AmbulanceData/' --The LOCATION clause specifies the location of the external tables. This clause is optional. USING 'odps-udf-example.jar'; --The USING clause specifies the Jar files that store the user-defined classes. This clause is optional, depending on whether you use user-defined classes.

csv または **tsv** ファイルに対し内蔵されているストレージハンドラーに対応するSTORED BY に続くパラメーターは次のとおりです。

com.aliyun.odps.CsvStorageHandler パラメーターは CSV 形式です。 CSV 形式
 でのデータの読み込みや書き込み方法を定義します。形式にはコンマ (,) で区切られてい

る列と行改行文字(\n)で終わる行が含まれます。たとえば、STORED BY'com.aliyun. odps.CsvStorageHandler'はサンプルパラメーターです。

 com.aliyun.odps.TsvStorageHandler パラメーターは、TSV形式です。TSV形式 でのデータの読み込みと書き込み方法を定義します。形式にはタブ文字 (\t) で区切られて いる列と行改行文字 (\n) で終わる行が含まれます。

STORED BY に続くパラメーターは、**TextFile、 SequenceFile、RCFile、 AVRO、 ORC、** および **Parquet** といった **open-source file formats** に対するストレージハンドラーの指 定もサポートします。 **TextFile** 形式では、**SerDe** クラスを指定します。 たとえば、org. apache.hive.hcatalog.data.JsonSerDe です。

- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as textfile
- org.apache.hadoop.hive.ql.io.orc.OrcSerde -> stored as orc
- org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe -> stored as parquet
- · org.apache.hadoop.hive.serde2.avro.AvroSerDe -> stored as avro
- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as sequencefi le

For external tables that are in the open-source formats, the statements to create tables are as follows.

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] (<column schemas>)
[PARTITIONED BY (partition column schemas)]
[ROW FORMAT SERDE '']
STORED AS
[WITH SERDEPROPERTIES ( 'odps.properties.rolearn'='${roleran}'
[,'name2'='value2',...]
) ]
LOCATION 'oss://${endpoint}/${bucket}/${userfilePath}/';
```

SERDEPROPERTIES 句の属性は次の表に示すとおりです。 現在、OSS で CSV および TST ファイルから gzip 形式で圧縮されたデータに対して、MaxCompute は内蔵のエクストラク ターを使った読み取りのみをサポートしています。 ファイルを gzip 圧縮にするかどうかを選 択します。 属性設定はファイル形式によって異なります。

属性	値	デフォルト値	説明
odps.text.option.gzip. input.enabled	true/false	false	圧縮データの読み取 りを有効、無効にし ます。

odps.text.option.gzip. output.enabled	true/false	false	圧縮データの書き込 みを有効、無効にし ます。
odps.text.option. header.lines.count	N (非負整数)	0	ファイルの最初の N 行をスキップしま す。
odps.text.option.null. indicator	String		NULL を文字列の値 に置換します。
odps.text.option. ignore.empty.lines	true/false	true	空白行を無視するか どうかを指定しま す。
odps.text.option. encoding	UTF-8/UTF-16/US- ASCII	UTF-8	ファイルのエンコー ド設定を指定しま す。

LOCATION 句は、外部テーブルの格納アドレスを指定します。形式は、oss://oss-cn-

shanghai-internal.aliyuncs.com/BucketName/DirectoryName です。 ダイアログ ボックスを使って OSS にディレクトリを選択します。 ファイルは選択しないでください。

[Tables] タブのノードディレクトリで DDL 文を使って作成したテーブルを検索します。 Level 1 Topic または Level 2 Topic を変更して、テーブルのディレクトリを変更します。

2. Table Store の外部テーブル

Table Store で外部テーブルを作成する文は次のとおりです。

```
CREATE EXTERNAL TABLE IF NOT EXISTS ots_table_external(
odps_orderkey bigint,
odps_orderdate string,
odps_custkey bigint,
odps_orderstatus string,
odps_totalprice double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler'
WITH SERDEPROPERTIES (
'tablestore.columns.mapping'=':o_orderkey,:o_orderdate,o_custkey,
o_orderstatus,o_totalprice', -- (3)
'tablestore.table.name'='ots_tpch_orders'
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole
'
```

```
LOCATION 'tablestore://odps-ots-dev.cn-shanghai.ots-internal. aliyuncs.com';
```

説明:

- com.aliyun.odps.TableStoreStorageHandler は MaxCompute ビルトインスト レージハンドラーで、Table Store のデータを処理します。
- ・ SERDEPROPERITES は、パラメーターのオプションを提供します。

TableStoreStorageHandler を使って「**tablestore.columns.mapping**」と 「**tablestore.table.name**」を指定する必要があります。

- tablestore.columns.mapping: このパラメーターは必要です。MaxCompute がアクセスする Table Store のテーブルの列を説明します。主キー列とプロパ ティ列を含みます。主キー列は、列名の前にコロン(:) で示します。この例で は、主キー列は、p:o_orderkey と :o_orderdate です。他は、プロパティ列 です。Table Store は最大 4 つの主キー列をサポートします。データタイプに は、String、Integer、Binary です。主キーの1番目の列は、パーティションキーで す。マッピングを指定する際、Table Store にあるテーブルのすべての主キー列を指定 します。MaxCompute がすべてのプロパティ列を指定する代わりにアクセスするプロ パティ列を指定する必要があります。
- tablestore.table.name: Table Store にアクセスするテーブル名 テーブル名が
 Table Store にない場合、エラーが報告されます。 MaxCompute は Table Store に
 テーブルを作成しません。
- ・ LOCATION: Table Store インスタンスの名前とエンドポイントを指定します。

3. GUI でテーブルを作成します。

[Data Development] ページへ移動し、「テーブルの管理」を参照し GUI でテーブルを作成 します。外部テーブルには次の属性が含まれます。

- ・基本的な属性
 - Table name (テーブルを作成し名前を入力します)
 - Table alias
 - Level 1 Topic と Level 2 Topic
 - Description
- · Physical model
 - Table type: 外部テーブルを選択します。
 - Partition: Table Store にある外部テーブルはパーティション化をサポートしません。
 - [Select the memory address]: LOCATION 句を指定します。次の図に示すとおり、
 LOCATION 句を [Physical model] セクションで指定します。ダイアログボックスで外部テーブルの格納場所をクリックし選択します。 [One-click authorization] を実行します。
 - [Select storage format]: 必要とされるファイル形式を選択します。
 CSV、TSV、TextFile、SequenceFile、RCFile、AVRO、ORC、および Parquet と カスタムファイル形式がサポートされています。 カスタムファイル形式を選択する場

- 合、対応するリソースを選択する必要があります。 リソースを送信すると、リソースからクラスが自動的に解析されます。 クラス名を選択します。
- rolearn: STS 権限が付与されている場合、この rolearn 属性を指定する必要はありません。
- Table structure design

Table Structure					
Add Field Move Up Move Down					
Field English Name Field Display Name	Field Type	Length or Settings	Description	Primary Key 🕐	Actions
age	bigint		**	No	
job	string		INSE	No	
marital	string		67	No	
education	string		经营销 用	No	
default	string		经发展性的利用	No	

- Data type: MaxCompute 2.0 ではフィールドに対し
 て、INYINT、SMALLINT、INT、BIGINT、VARCHAR および STRING タイプを サポートしています。
- Actions: フィールドを作成、変更、削除します。
- Length/Set: VARCHAR タイプの列の最大長を指定します。 コンポジットデータタイ プでは、定義を入力することができます。

サポートしているデータタイプ

外部テーブルでサポートされている基本データタイプは次の表に示すとおりです。

データ型	新規	例	説明
TINYINT	न्	1Y, -127	署名された 8 ビットの整数で、範 囲は -128 から 127です。
SMALLINT	न्	327678, -1008	署名された 16 ビットの整数で、範 囲は -32,768 から 32,767 です。
INT	न]	1000, -15645787	署名された 32 ビットの整数で、範 囲は -231 から 231-1 です。
BIGINT	不可	100000000000L, -1L	署名された 64 ビットの整数で、範 囲は -263 + 1 から 263 - 1 です。
FLOAT	न]	なし	32 ビットのバイナリ浮動小数点で す。

DOUBLE	不可	3.1415926 1E+7	8 ビットの倍精度浮動小数点です (64 ビットバイナリ浮動小数点)
DECIMAL	不可	3.5BD, 9999999999999. 9999999BD	10 進の抽出数値です。 精度範囲 は-1036 + 1 から 1036 -1で、ス ケールは 10 から 18 です。
VARCHAR(n)	ग	なし	変数長の文字列です。 長さは n で す。範囲は、1 から 65535 です。
STRING	不可	"abc"、'bcd '、"alibaba"	文字列 現在、最大長は 8M です。
BINARY	ग	なし	バイナリ数値です。 現在、最大長 は 8M です。
DATETIME	不可	DATETIME '2017- 11-11 00:00:00'	日時と時間のデータタイプです。 UTC-8 はシステムの標準時間と して使用されています。 範囲は 0000-01- 01 から 9999-12-31 で、ミリ秒で正確です。
TIMESTAMP	न्	TIMESTAMP '2017 -11-11 00:00:00. 123456789'	TIME STAMP データタイプで、 タイムゾーンと独立したもので す。 範囲は 0000-01- 01 から 9999-12-31 で、ナノ秒で正確で す。
BOOLEAN	不可	TRUE、 FALSE	論理ブール (TRUE/FALSE)

外部テーブルでサポートされているコンポジットデータタイプは次の表に示すとおりです。

タイプ	説明	コンストラクター
ARRAY	array< int >; array< struct < a:int, b:string >>	array(1, 2, 3); array(array(1, 2); array(3, 4))
МАР	map< string, string >; map < smallint, array< string>>	map("k1", "v1", "k2 ", "v2"); map(1S, array ('a', 'b'), 2S, array(' x', 'y))
STRUCT	struct< x:int, y:int>; struct < field1:bigint, field2: array< int>, field3:map< int, int>>	named_struct('x', 1, ' y', 2); named_struct(' field1', 100L, 'field2', array(1, 2), 'field3', map (1, 100, 2, 200))

MaxCompute 2.0 で新しくサポートされたデータタイプ (TINYINT、SMALLINT、 INT、 FLOAT、VARCHAR、TIMESTAMP、BINARY またはコンポジットデータタイプ) を使用す

る場合、テーブルを作成する文の前に set odps.sql.type.system.odps2=true; を含めま す。文を送信、実行し、設定文を使ってテーブルを作成します。 HIVE との互換性が必要な場 合、odps.sql.hive.compatible=true; 文を含めます。

外部テーブルの表示と処理

Tables ビューで外部テーブルを探します。



外部テーブルの処理は、内部テーブルの処理と似ています。外部テーブルに関する詳細は、 「テーブルの管理」と「#unique_64」をご参照ください。

1.16 関数

関数の一覧では、現在利用可能な関数、関数の分類、関数の使用説明およびインスタンスを提供 します。

関数の一覧には6つの部分があり、その他の関数、文字列処理関数、数学関数、日付関数、ウィ ンドウ関数、および集約関数が含まれています。 これらの関数はシステムによって提供されてい ます。 関数をドラッグして、その説明と使用方法を表示します。

	Functions	C
 (/)	Enter a function name	Q
*	✓ ■ Functions	
R	> 🛅 String function	
-	> 🛅 Other function	
	> 🛅 Mathematical Function	
	> 🛅 Date function	
#	> 🛅 Analytic function	
	 Aggregate function 	
5	Fx avg	
£×	Fx count	
	Fx avg	<
Ū	Ev may	
	Description	
	 command format: avg(value) <u>n</u> bsp; 	: 1.5

1.17 **エディターショートカット一**覧

コード編集用の共通ショートカット

Windows クロームバージョン

- Ctrl + S保存
- Ctrl + Z元に戻す
- Ctrl + Y やり直し
- Ctrl + D同じワードを選択

Ctrl + X行の切り取り

Ctrl+Shift+K 行の削除

Ctrl + C 現在の行をコピー

Ctrl+i 行を選択

- Shift+Alt+マウスをドラッグ列モード編集、この部分のすべてのコンテンツを編集
- Alt + マウス 複数列モード編集、マルチラインインデント
- Ctrl + Shift + L 同一のすべての文字列インスタンスにカーソルを追加、バッチ変更
- Ctrl + F 検索
- Ctrl + H 置換
- Ctrl + G 特定の行を検索
- Alt + Enter 検索で一致するすべてのキーワードを選択
- Alt↓ / Alt↑ 現在の行を上下に移動
- Shift + Alt + ↓ / Shift + Alt + ↑現在の行を上下にコピー
- Shift + Ctrl + K現在の行を削除
- Ctrl + Enter / Shift + Ctrl + Enter カーソルを上下に移動
- Shift + Ctrl + \ 一致するブラケットへカーソルを移動
- Ctrl +] / Ctrl + [インデントの追加/削除
- Home / End 現在の行の先頭または終わりへ移動
- Ctrl + Home / Ctrl + End 現在のファイルの先頭/終わりへ移動
- Ctrl + → /Ctrl + ← 単語ごとにカーソルを左右へ移動
- Shift + Ctrl + [/ Shift + Ctrl +] カーソルでポイントしたブロックを表示/非表示
- Ctrl + K + Ctrl + [/ Ctrl + K + Ctrl +] カーソルでポイントしたサブブロックを 表示/非表示
- Ctrl + K + Ctrl + 0 / Ctrl + K + Ctrl + j すべてのエリアを折りたたむ/展開する
- Ctrl + / カーソルがある行またはコードブロックにコメントを書き込む/消去する

MAC クロームバージョン

- cmd + S保存
- cmd + Z 元にもどす
- cmd + Y やり直し

- cmd+D同じワードを選択
- cmd + X 行の切り取り
- cmd + shift + K行の削除
- cmd + C 現在の行をコピー
- cmd +i 現在の行を選択
- cmd + F 検索
- cmd + alt + F 置換
- alt↓ / alt↑ 現在の行を上下に移動
- shift + alt + ↓ / shift + alt + ↑現在の行を上下にコピー
- shift + cmd + K 現在の行を削除
- cmd + Enter / shift + cmd + Enter カーソルを上下に移動
- shift + cmd + \ 一致するブラケットへカーソルを移動
- `cmd +] / cmd + [インデントの追加/削除
- cmd + ← / cmd + → 現在の行の先頭/終わりへ移動
- cmd + ↑ / cmd + ↓ 現在のファイルの先頭/終わりへ移動
- alt + → /alt + ← 単語ごとにカーソルを左右に移動
- alt + cmd + [/ alt + cmd +] カーソルでポイントしているブロックを表示/非表示
- cmd + K + cmd + [/ cmd + K + cmd +] カーソルでポイントしているサブブロックを表示/非表示
- cmd + K + cmd + 0 / cmd + K + cmd + j すべてのエリアを折りたたむ/展開する
- cmd + / カーソルがある行またはコードブロックにコメントを書き込む/消去する

複数カーソル/選択

- alt + Clicking with the mouse カーソルの挿入
- alt + cmd + ↑/↓ カーソルを上下に挿入
- cmd + U 最後のカーソル操作を元に戻す
- shift + alt + Iカーソルを選択したコードブロックの各行の最後に挿入

cmd + G/shift + cmd + G次/前の項目を検索
cmd + F2マウスが選択したすべての文字を選択
shift + cmd + Lマウスが選択したすべての部分を選択
alt+Enter 検索で一致したキーワードをすべて選択
shift + alt + マウスをドラッグ編集する複数の列を選択
shift + alt + cmd + ↑ / ↓編集する複数の列を選択するため、カーソルを上下に移動
shift + alt + cmd + ← / →編集する複数の列を選択するため、カーソルを左右に移動

1.18 ゴミ箱

DataWorks には独自のゴミ箱があります。ページの左上隅にある [Recycle Bin] をクリックします。



[Recycle Bin] ページでは、現在のプロジェクトで削除されたすべてのノードを確認することが できます。 ノードを右クリックして、元に戻す、または完全に削除することができます。

[Recycle Bin] ページの右側にある [Show My Files] をクリックして、削除されたノードを表示 します。



ノードがゴミ箱から完全に削除された場合、元に戻すことはできません。

2 DataService Studio

2.1 DataService Studio の概要

DataService Studio は、企業がプライベートおよびパブリックの API を集中管理できるよう、 データサービスのバスを構築することを目的としています。 DataService Studio を使用するこ とで、データテーブルに基づいた API をすばやく作成することができます。また、一元的な管 理とリリースを目的として、DataService Studio プラットフォームに既存の API を登録する ことができます。なお、DataService Studio は API Gateway に接続されています。 ワンク リックで API Gateway に API をデプロイすることができます。 DataService Studio は、API Gateway と連携することで、安全かつ安定した、低コストで使いやすいデータの共有サービス を提供します。

DataService Studio はサーバーレスアーキテクチャーを採用しています。 ユーザーは、実行 環境などのインフラストラクチャなどに気を配る必要はなく、API のクエリロジックを気にする 必要があるだけです。 DataService Studio では、ユーザー用のコンピューティングリソースを 準備しています。また、弾力的なスケーリングをサポートしており、O&M コストがかかりませ ん。

データ API の作成

現在、DataService Studio では、リレーショナルデータベースおよび NoSQL データベース のテーブルに基づいたデータ API をすばやく作成するため、視覚化されたウィザードの使用を サポートしています。コードを記述することなく、数分でデータ API を設定することができま す。高度なユーザーの、パーソナライズされたクエリ要件を満たすため、DataService Studio では、カスタマイズ SQL スクリプトモードを提供しています。この機能により、 API クエリの SQL 文をユーザー自身でコンパイルすることができます。また、複数テーブルの関連付け、複雑 なクエリ条件、集計関数もサポートされています。

API の登録

DataService Studio では、DataService Studio で登録した既存の API サービス、および データテーブルに基づいて作成された API の集中管理もサポートしています。 現在登録でき るのは RESTful API のみになります。 サポートされているリクエストメソッドには、GET、 POST、PUT、および DELETE があります。 サポートされているデータ型は、JSON 形式デー タと XML 形式データです。
API Gateway

API Gateway では、API のパブリッシュ、管理、メンテナンス、および API サブスクリプション期間の管理などの API 管理サービスを提供します。 API Gateway では、パートナーや開発者を対象として、マイクロサービスの統合、フロントエンドとバックエンドの分離、およびシステム統合を実装するための、シンプルかつ高速で低コスト低リスクのメソッドを提供します。

DataService Studio は API Gateway に接続されています。 管理を行うため、ユーザーは、 API の権限付与と認証、トラフィック制御、メータリングなど、DataService Studio で作成お よび登録されたすべての API を API Gateway にデプロイすることができます。

API Market

Ali cloud の API market は、金融、人工知能、電子商取引、交通地理学、生活サービス、企業管理、および 8 つのパブリックアフェアーズ主要カテゴリーを網羅する、中国で最も総合的な API 売買市場です。ここでは、数千の API 製品がオンラインで販売されています。

ユーザーの各種 API を、DataService Studio から API Gateway にパブリッシュした後、そ れらを Alibaba Cloud API Marketplace に公開することができます。 この方法により、ユー ザーの会社は経済的利益を簡単に得ることができます。

2.2 用語集

データサービスに関連する単語について、以下に説明します。

- ・データソース:データベースのリンクです。データサービスでは、データソースを経由して
 データにアクセスします。データソースは Data Integration 内でのみ、設定を行えます。
- ・ API の作成: データテーブルに基づいて API を作成します。
- ・ API の登録: 統合管理用に既存の API を Data Service に登録します。
- ・ ウィザード: API 作成の手順を案内します。 このメソッドはシンプルな API を作成したい初心 者に適しています。 ユーザーはコードを記述する必要はありません。
- ・スクリプト: SQL スクリプトを記述し、API を作成します。このメソッドは、テーブル結合クエリ、複雑なクエリ、集計関数をサポートしています。このメソッドは、複雑な API を作成したい経験豊富な開発者に適しています。
- API グループ: API グループは、特定のシナリオまたは特定のサービスを利用するための API セットです。 API グループは、データサービス内での最小のグループ単位、および API Gateway によって管理される最小の単位です。 API グループは、API 製品として Alibaba Cloud API Market に公開されています。

- API Gateway: API を管理するため、Alibaba Cloud によって提供されるサービスです。
 API Gateway は、API のサブスクリプション期間管理、権限管理、アクセス管理、およびトラフィック制御をサポートしています。
- API Market: Alibaba Cloud API Market は、Alibaba Cloud Market 上に設置された、
 国内で最も完璧かつ統合的な API 取引のプラットフォームです。

2.3 API **の生成**

2.3.1 データソースの設定

データ API を使用してサービスを生成する前に、データソースを事前に設定する必要がありま す。 データサービスを使用することで、データソースからデータテーブルのスキーマ情報を取得 し、API リクエストを処理することができます。

Dataworks コンソールの [data integration] > [data source] ページにて、データソースを設 定します。各種データソースタイプのサポートおよびそれらの設定方法を次の表に示します。

データソース名	データ API 生	データ API 生	設定方法
	成のウィザード	成のスクリプ	
	モード	トモード	
RDS (ApsaraDB for RDS)	対応	対応	RDS には、MySQL、 PostgreSQL、および SQL Server が含まれています。
DRDS	対応	対応	#unique_73
MySQL	対応	対応	#unique_74
PostgreSQL	対応	対応	#unique_75
SQL Server	対応	対応	#unique_76
Oracle	対応	対応	#unique_77
AnalyticDB (ADS)	対応	対応	#unique_78
Table Store (OTS)	対応	非対応	#unique_79
MongoDB	対応	非対応	#unique_80

2.3.2 API 生成の概要

現在、データサービスでは、視覚的に設定されたウィザードモードにより、リレーショナルデー タベースおよび NoSQL データベースによる、テーブルの高速生成をサポートしています。デー タ API では、データ API を設定するため、ものの数分でプログラムを書くような能力は必要あ りません。 高度なユーザーの、パーソナライズされたクエリ要件を満たすため、データサービス にはカスタマイズ SQL スクリプトモードが用意されています。これにより、API クエリの SQL 文を自身でコンパイルですることができます。 また、複数テーブルの関連付け、複雑なクエリ条 件、集計関数もサポートされています。

機能	特徴	ウィザードモード	スクリプトモード
オブジェクトの照 会	1 つのデータソースから、 1 つのデー タテーブルを照会します。	対応	対応
	1 つのデータソースから、複数の結 合テーブルの照会	非対応	対応
フィルターバー	正確な数値を照会します。	対応	対応
	数値範囲の照会	非対応	対応
	正確な文字列との突き合わせ	対応	対応
	文字列のあいまい検索	対応	対応
	必須 / オプションのパラメーターの 設定	対応	対応
クエリの結果	フィールドの値を返します。	対応	対応
	フィールド値の数学的計算結果を返 す	非対応	対応
	フィールド値の集計計算結果を返す	非対応	対応
	ページネーションによる結果の表示	対応	対応

ウィザードモードおよびスクリプトモードの機能は、以下のとおりです。

2.3.3 ウィザードモードでの API の生成

本ページでは、ウィザードモードでの API 生成の手順および考慮事項について説明します。

ウィザードモードを使用してデータを生成すると、コードを記述することなく、簡単に API を使 用することができます。プロダクトインターフェイスから設定を確認することにより、API をす ばやく生成することができます。 API の機能に対する高度な要件を持たないユーザー、または コード開発経験の少ないユーザーは、このウィザードを使用することを推奨します。

注:

API の設定を行う前に、Dataworks コンソールの[Data integration] > [Data Source] ペー ジにて、データソースを設定します。

API 基本情報の設定

- 1. [API Service list] > [Generate API] に移動します。
- 2. [Wizard Mode] をクリックし、API の基本事項を入力します。

1 API Ba	sic Information —	(2	API Parameters	API Testing	Next
* API Name	test_API				
	Support Chinese cho	aracters, English, numbers,	underline, and must start with English	h or Chinese characters, 4 to 50 characters	
* API Group	WorkShop	→ + Add AP	1 Group		
+ Protocol	🖌 HTTP				
 API Path 	/api/demo				
	Support for English,	number, underscore, hyphe	ns (-), and must start with /, not more	e than 200 characters, etc: /user	
Request	GET	×			
	1001				
Response	JSUN	~			
Description	API demo				

設定では、 API グループ化の設定に注意してください。 API グループには、特定のシナリオ に使用される各種 API 集が含まれています。 これは API Gateway での最小管理単位です。 Alibaba Cloud API マーケットでは、各 API グループは特定の API に対応しています。

🧾 注:

API グループ化の設定例は次のとおりです。

たとえば、天気を照会するのに API プロダクトの設定を行いたい場合、都市名検索による天 気検索 API、観光地天気検索 API、郵便番号検索による天気 API の 3 種類の API プロダク トを設定し、"天気の照会" と名付けた API グループを作成し、このグループに上記の 3 つの API を配置します。 その API がマーケットに公開されると、天気の照会プロダクトとして表 示されます。

もちろん、ユーザーが生成した API がユーザー自身のアプリ内で使用される場合、分類するのにグループ化を行うことができます。

現在、ビルド API は HTTP プロトコル、GET リクエストモード、および JSON リターンの みをサポートしています。

3. API の基本情報を入力したら、[Next] をクリックし、API パラメーター設定ページに移動し ます。

API パラメーターの設定

1. [Data source type] > [Data source name] > [Table] に移動し、設定するテーブルを選択 します。

首注:

データセット内でデータソースを事前に設定する必要があります。データテーブルのドロッ プダウンボックスは、テーブル名検索に対応しています。

2. 次に、リクエストとレスポンスのパラメーターを指定します。

データテーブルが選択されると、テーブルのすべてのフィールドが左側に表示されます。 リク エストパラメーターとレスポンスパラメーターとして使用するフィールドを選択し、それらを 対応するパラメーターリストに追加します。

3. 最後に、パラメーター情報を編集して完了です。

リクエストおよびリターンパラメータリスト右上の、[Edit]をクリックし、パラメーター情報 の編集ページに移動します。パラメーターの名前、サンプル値、既定、必須、あいまい一致 (文字列タイプのみサポート)の設定を行います。オプションと説明のフィールドは必須です。

D ∣ Ge	nerate API		(API Basic Inform	nation		💿 API	Parameters		- 🧿 API Testir	0		Back	Next	2
Configura	tion Table	MySQL	✓ da,wokah	tp.jog	~	region	,day,ptat	×							
Configura	tion Paramet	ers Field	Search Parameter Field Q			Request	parameter								Edit
	Field		Field Type	Index			Parameter Name	Binding Field	Type	Example Value	Default Value	Required	Fuzzy Matching	Description	
	biadete		DATE				region	region	string			Yes	No		
	region (REQ)		VARCHAR												
	ри		BIGINT												
0	UM .		BIGINT												
	browse_size (R	ES	BIGINT												
						Respons	e parameter 🕕	Response Pagination	on .						Edit
							Parameter 3	lame	Binding Field	η	pe.	Example Value	Descriptio		
							browse_size		browse_size	lo	ng				

設定プロセスでは、ページング結果の戻り設定に注意を払う必要があります。

- [Response pagination] を有効にしない場合、既定では、API は最大 500 レコードを出力 します。
- ・ 返される結果が 500 を超える可能性がある場合、[Response pagination] 機能をオンにします。

次のパブリックパラメーターは、[Response pagination] 機能が有効になっている場合にの み、使用できます。

- ・ 共通のリクエストパラメーター
 - pageNum: 現在のページ番号です。
 - Pagesize: ページサイズです。 つまり、1 ページごとのレコード数です。
- ・ 共通のレスポンスパラメーター
 - pageNum: 現在のページ番号です。
 - Pagesize: ページサイズです。 つまり、1 ページごとのレコード数です。
 - totalNum: レコードの合計数です。

注:

- ・ リクエストパラメーターは、同等のクエリのみをサポートし、リターンパラメーターは、現状の値のフィールド出力のみをサポートします。
- 可能な限り、インデックスフィールドをリクエストパラメーターに設定します。
- API にはリクエストパラメーターを指定することはできません。その場合、ページネーション機能を有効にする必要があります。
- ・ API の呼び出し元にとって、API の詳細を理解しやすくするため、API のサンプル値、既定 値、および説明パラメーターを指定することを推奨します。
- 現在のテーブルで生成されている API のリストを表示するには、構成済みの API をクリック します。同じ API が生成されないようにします。

API パラメーターの設定が完了したら、[Next] をクリックし、API のテストセクションに移動 します。

API のテスト

API パラメーターの設定が完了したら、API のテストを開始します。

		API Service Test	
81			
88	API Service Test	test,API V GET JSON	Request Details
0		APLPRTH: /wpi/demol	ateri to test ap(462) test, API
		Request Parameter	Overy parts test case parameters. (OX)
		Parameter Name Parameter Type Required Value	build backend ool applequent. (XX) apple backend ool applequent. (XX) apple ool templete: SELECT browse, alor: AS browse, alor, bizdete AS bizdete, pv AS pv; vv AS uv FROM region, dey, attet W
		region string Yes Designg	n Kur, tegon = r ani soj parametera: [bojing] resty to execute polybojeet. (DC] andiperater terancia fendande (DC)
			Response Content
			1 6 *444* D. 3 ****Cat* 0, 6 *****Cat*: 5 ****generid*: ***********************************
			Test Successfully API Call delay : 19 ms

パラメーターを設定し、[Start Test] をクリックして、API リクエストをオンラインで送信しま す。API リクエストの詳細と応答結果が右側に表示されます。 テストが失敗した場合は、エラー メッセージを注意深く読み、適切な調整を行って API を再テストします。

設定プロセスでは、標準の応答例の設定に注意する必要があります。 API をテストするとき、シ ステムは自動的にエラー例およびエラーコードを生成します。 ただし、標準の応答例は自動的に は生成されません。 テストが成功したら、[Save as Normal Response Sample] をクリック し、今のテスト結果を通常の応答サンプルとして保存します。 機密データが応答に含まれている 場合、手動で編集することができます。

🗇 👔 Generate API		API Basic Information (V API Parameters (V API Testing Back Fruch
ShowAPI GET JSON		Request Details
API PATH : /spi/demo3		
Request Parameter		Query
Parameter Name	Parameter Type	1 1
pageNum	in	2 "data": { 3 "totalNum": 1,
pageSize	jec.	<pre>% pageblas 1 40, 5 "rows": [{ 6 "name" * 0137a9699b3c927b587bdd9c1568a5aff1ad31d4fc83f806b284c2b25</pre>
		<pre>} }, ***********************************</pre>

▋ 注:

- ・標準の応答例は、API 呼び出し元に重要な参照値を提供します。可能であれば例を明記します。
- ・ API 呼び出し遅延は、現在の API リクエストの遅延です。これは、API のパフォーマンスを 評価する際に使用されます。 レイテンシが高すぎる場合は、データベースの最適化を検討し てください。

API のテストが完了したら、[Finish] をクリックします。 以上により、API は正常に作成されます。

API 詳細の閲覧

[API Service list] ページに戻り、操作列の [details] をクリックし、API の詳細を表示しま す。 このページには、呼び出し元から見た API に関する詳細情報が表示されます。

ち + API Service Details							Status : Draft	API Service Test
ß ^{gr} test_API								
i≣ API Basic Information ∨	Request Parameters							^
API ID 462	Application-level request	parameters						
Principal auglin	Parameter Name	Type	Example Value	Default Value	Required	Fuzzy Metching	Description	
Create Time 2018-09-04 15:57:13 Description API demo	region	string			Yes	No		
HTTP API Info								
HTTP API ad http://do-server.cn-shanghai.deta.al	Request Parameter							^
dress iyun-inc.com/project/79023/api/de mo1	 Application-level response 	e parameters						
Request GET Response JSON	Parameter Name		Type	Example Va	lue	Description		
Data Source Information ~	browse_size		long					
Name rds_workshop_log	bizdete		string					
Type mysql	pv		long					
Connection JDBC UH jellen mynagl if 7 BE 108.84.1 TH87/wo	UV		long					
Username workshop								
Table Name region_dey_stat	Correct Response Exam	ple						^
Description rds log data syc								

2.3.4 スクリプトモードでの API の生成

本ページでは、スクリプトモードでの API 生成の実行手順を説明します。

パーソナライズされたクエリに対するハイエンドユーザーの要件を満たすため、データサービス では SQL をカスタマイズするためのスクリプトパターンも提供しています。これにより、API のSQL クエリを自分自身で作成することが可能になります。複数テーブルの関連付け、複雑なク エリ条件、および集約関数に対応しています。

API 基本情報の設定

1. [API Service list] > [Generate API] に移動します。

2. [Script Mode] をクリックし、API の基本事項を入力します。

1 API Bas	iic Information 2 API Parameters 3 API Testing
* API Name	test_API
	Support Chinese characters, English, numbers, underline, and must start with English or Chinese characters, 4 to 50 characters
* API Group	WorkShop 💛 + Add API Group
* Protocol	
* API Path	/api/demo
	Support for English, number, underscore, hyphens (-), and must start with /, not more than 200 characters, etc: /user
Request	GET 🗸
Response	JSON
Description	
 Description 	API demo

設定では、 API グループ化の設定に注意してください。 API グループには、特定のシナリオ に使用される各種 API 集が含まれています。 これは API Gateway での最小管理単位です。 Alibaba Cloud API マーケットでは、各 API グループは特定の API に対応しています。

_____注:

API グループ化の設定例は次のとおりです。

たとえば、天気を照会するのに API プロダクトの設定を行いたい場合、都市名検索による天 気検索 API、観光地天気検索 API、郵便番号検索による天気 API の 3 種類の API プロダク トを設定し、"天気の照会" と名付けた API グループを作成し、このグループに上記の 3 つの API を配置します。 その API がマーケットに公開されると、天気の照会プロダクトとして表 示されます。

もちろん、ユーザーが生成した API がユーザー自身のアプリ内で使用される場合、分類するのにグループ化を行うことができます。

現在、ビルド API は HTTP プロトコル、GET リクエストモード、および JSON リターンの みをサポートしています。

3. API の基本情報を入力したら、[Next] をクリックし、API パラメーター設定ページに移動し ます。

API パラメーターの設定

1. データソースとテーブルを選択します。

[Data source type] > [Data source name] > [Data Table] に移動します。データテーブ ルリスト内の適切なテーブル名をクリックし、このテーブルのフィールド情報を表示します。

▋ 注:

- ・データセット内でデータソースを事前に設定する必要があります。
- ・データソースを選択する必要があります。データソース間のテーブル結合クエリには対応 していません。
- 2. API の SQL クエリを書きます。

右側のコードボックスに SQL コードを入力することができます。 システムは、フィー ルドのリスト中のフィールドをチェックするワンクリックの SQL 機能に対応していま

す。[Generate SQL] をクリックすると、SELECT xxx FROM xxx の SQL 文が自動的に生成 され、右カーソルに挿入されます。

ち i Generate API	API Basic Information ————————————————————————————————————	API Parameters API Testing	Back Next I 🖾
MySQL V	SQL Statement		Code Parameter
mysqLrds 🗸 🗸	<pre>1 SELECT * FROM mysql_rds;</pre>		
Search Table Name Q			昌
Table Name DB Name Description			
aa mysql_rds			
test mysql_rds			
px_31 mysql_rds			
person mysql_rds			
<			
Field Name Type Description			
No data			
1			
1			



- ・ ワンクリックの SQL 追加は、フィールド数が比較的多い場合に特に便利です。これにより、SQL 作成の効率が大幅に向上します。
- SELECT クエリのフィールドは、API のリターンパラメーターです。where 条件のパラメーターは、API のリクエストパラメーターです。リクエストパラメーターは "\$" で識別されます。

3. 最後に、パラメーター情報を編集して完了です。

API クエリの SQL を作成した後、右上の [parameters]をクリックし、パラメーター情報の 編集ページに切り替えます。編集ページでは、タイプ、サンプル値、既定値、およびパラメー ターの説明を編集します。タイプとパラメーターの説明の編集は必須です。

🧵 注:

API の呼び出し者が API をより総合的に理解できるようにするため、可能な限り、API パラ メーター情報を入力します。

5 Generate API	API Basic Inform	nation	2 API Parameters	3 API Testing		Back	Nest 1 🔛
MySQL	 API Parameters 						Code Parameter
mysql_rds	· Remark Parameter						
Search Table Name	Q						
Table Name DB Name Descr	Parameter Name	Type	Example Value	Default Value	Description		
aa mysql_rds							
teat mysql_rds							
px_31 mysql_rds							
person mysql_rds							
<u>.</u>							
	Response Parameter	Response Pagination					
Field Name Type Des	scription						
🖌 id INT	Parameter Name	Туре	Example Value	Description			
🗹 name VAR							
sex TINY							
salary BIGL.							
()	2						

設定プロセスでは、ページング結果の戻り設定に注意を払う必要があります。

- [Response pagination] を有効にしない場合、既定では、API は最大 500 レコードを出力 します。
- ・ 返される結果が 500 を超える可能性がある場合、[Response pagination] 機能をオンにします。

次のパブリックパラメーターは、[Response pagination] 機能が有効になっている場合にの み、使用することができます。

- ・ 共通のリクエストパラメーター
 - pageNum: 現在のページ番号です。
 - Pagesize: ページサイズです。 つまり、1 ページごとのレコード数です。

・ 共通のレスポンスパラメーター

- pageNum: 現在のページ番号です。
- Pagesize: ページサイズです。 つまり、1 ページごとのレコード数です。
- totalNum: レコードの合計数です。

_____注:

SQL ルールプロンプト

- 対応している SQL 文は1つだけであり、複数の SQL 文には対応していません。
- 対応しているのは、SELECT クラスだけです。INSERT、UPDATE、DELETE などの他の クラスは対応していません。
- ・ SELECT クラスを対象としたクエリフィールドは、API のリターンパラメーターです。 where 条件内の **\$ {Param}** の変数 **"Param"** は、API のリクエストパラメーターです。
- · SELECT * はサポートされていません。クエリの列は明示的に指定する必要があります。
- ・単一のテーブルクエリ、テーブル結合クエリ、および単一のデータソース内のネストされた
 クエリがサポートされています。
- SELECT クエリ列の列名に、テーブル名のプレフィクス (たとえば、T. name など) がある 場合、このエイリアスをリターンパラメーター名 (たとえば、T. name を名前とする) と見な す必要があります。
- ・集計関数 (min / max / sum / countなど) を使用する場合は、エイリアスをリターンパラメーター名として使用する必要があります (total \ _ num を sum (Num) とする)。
- SQL では、リクエストパラメーターが置き換えられるとき、\$ {Param} は一様で、文字列 }
 に \$ {Param} を含んでいます。 \$ {Param} にエスケープ文字 \ があるとき、通常の文字列
 として処理されるリクエストパラメーター処理を行いません。
- たとえば、'\$ {ID}'、 'ABC \$ {xyz} 123' のように、 \$ {Param} を引用符で囲むことはサポー
 トされていません。 'abc', '\$ {xyz}', '123' の文字列の結合については、必要であれば実装す
 ることができます。

API パラメーターの設定が完了したら、[Next] をクリックし、API のテストセクションに移動 します。

API のテスト

API パラメーターの設定が完了したら、API のテストを開始します。

81		API Service Test	
83	API Service Test	NULAR V GET JON	Request Details
•		APLENT : Aplidenal APLENT : Aplidenal Report Parameter Parameter Parameter Report ething Yes Beging	Corey Werk you can serve any [44:3] tour, AP1 Werk you werk you can serve any [44:3] tour, AP1 Werk you can serve any many the provide any the
Γ			Test Successfully API Call delay : 19 ma

パラメーターを設定し、[Start Test] をクリックして、API リクエストをオンラインで送信しま す。API リクエストの詳細と応答結果が右側に表示されます。 テストが失敗した場合は、エラー メッセージを注意深く読み、適切な調整を行って API を再テストします。

設定プロセスでは、標準の応答例の設定に注意する必要があります。 API をテストするとき、シ ステムは自動的にエラー例およびエラーコードを生成します。 ただし、標準の応答例は自動的に は生成されません。 テストが成功したら、[Save as Normal Response Sample] をクリック し、今のテスト結果を通常の応答サンプルとして保存します。 機密データが応答に含まれている 場合、手動で編集することができます。

🗇 🗇 Generate API		0	API Basic Information	🕢 API	Parameters	🜖 API Testing	Beck	Finish
ShowAPI GET JSON					Request Details			
Request Parameter				Ouerv				
Property Name	December 7.000			4444)				
Parameter Name	Parameter type	1 1						
pageNum	int	¥ 3	"totalNum": 1,					
pageSize	jet.	¥ 5	"rows": [{					
		9 9 10 11 12 13 14 3	<pre>}; "olisiasourc }; "pageNum": 1 "errCode": 0, "errMsg": "success", "requestId": "188bd3d4-5e</pre>	15a-4193-a726-	ebaae3312087*	548a5aff1ac33d4f08 c04769137blcf16bb2	0580462846285 9739167a8ce3c	

📋 注:

・標準の応答例は、API 呼び出し者に重要な参照値を提供します。 可能であれば例を指定します。

・ API 呼び出し遅延は、現在の API リクエストの遅延です。これは、API のパフォーマンスを 評価する際に使用されます。 レイテンシが高すぎる場合は、データベースの最適化を検討し てください。

API のテストが完了したら、[Finish] をクリックします。 これで、データ API は正常に作成されました。

API 詳細の閲覧

[API Service list] ページに戻り、操作列の [details] をクリックし、API の詳細を表示しま す。 このページには、呼び出し者から見た API に関する詳細情報が表示されます。

ち I API Service Details							Status : Draft	API Service Test
,6 ⁰ test_API								
i≣ API Basic Information ~	Request Parameters							^
APIID 462 API Group WorkShop Principal swallin Create Time 2018-09-0415-57:13 Description API demo	Application-level request parameters							
	Parameter Name	Type	Example Value	Default Value	Required	Fuzzy Matching	Description	
	region	string			Yes	No		
HTTP API Info								
HTTP API ad http://do-server.on-changhai.data.al diess iyunino.com/project/79023/spi/de mo1 Request GET Request GET	Request Parameter							^
	 Application-level response 	parameters						
	Parameter Name		Type	Example Vi	slue	Description		
Data Source Information Name rds_workshop_log Type mysol Connection JOBC (M) jalaxempsel/1788 100.84.1 17857/wo	browse_size		long					
	bizdete		string					
	pv		long					
	UV		long					
dishap Username workshop								
Table Name region_day_stat Description rds log data syc	Correct Response Examp	sie						^

2.4 API の公開

API Gateway は、完全なライフサイクル管理を提供する API ホスティングサービスですライフサ イクル管理では、API のリリース、管理、O&M、販売をカバーしています。 API Gateway で は、パートナーや開発者を対象として、マイクロサービスの統合、フロントエンドとバックエ ンドの分離、およびシステム統合を実装するための、シンプルかつ高速で低コスト低リスクのメ ソッドを提供します。

API Gateway では、権限管理、トラフィック制御、アクセス制御、およびメータリングサービ スを提供します。 これらのサービスにより、API の作成、モニタリング、保全を簡単に行うこと ができます。 そのため、データサービスで作成および登録された API を API Gateway に公開 することを推奨します。 データサービスと API Gateway は接続されているため、各種 API を API Gateway に簡単に公開することができます。

各種 API の API Gateway への公開

📋 注:

API をリリースするには、まず API Gateway サービスを有効にする必要があります。

API Gateway を有効にした後、[API Service list] の操作列で、[Publish] をクリックし、API を API Gateway にリリースします。 システムは、パブリッシュプロセス中に、API を API Gateway に自動的に登録します。 システムは、API グループと同じ名前でグループを API Gateway に作成し、そのグループに API をリリースします。

リリース後、API Gateway コンソールにアクセスし、API 情報を表示します。 API Gateway では、スロットルおよびアクセス制御機能を設定することもできます。

API をアプリケーションから呼び出す必要がある場合、API Gateway でアプリケーションを作成し、そのアプリケーションに対して API を許可し、AppKey と AppSecret を使用して、署名の呼び出しを暗号化する必要があります。詳しくは、「API Gateway help documentation」をご参照ください。同時に、API Gateway では主流のプログラミング言語で開発ができる SDK の提供もしており、自分のアプリケーションに対して、自分が作成した API をすばやく統合することができます。詳しくは、「SDK のダウンロードとユーザーガイド」をご参照ください。

各種 API の Alibaba Cloud API マーケットプレイスへの公開

ユーザーの各種 API を、データサービスから API Gateway にパブリッシュした後、それらを Alibaba Cloud API Marketplace に公開することができます。 この方法により、ユーザーの 会社は経済的利益を簡単に得ることができます。

API を Ali cloud の API マーケットに販売する前に、まず最初に、サービスプロバイダーとし て Ali cloud の API market にログインする必要があります。

注:

次の図に示すように、API Marketplace への移動を選択します。 注記: Alibaba Cloud API Marketplaceに参加することができるのは、エンタープライズユーザーのみになります。

手順

- 1. Ali cloud の「サービスプロバイダープラットフォーム」に移動します。
- **2.** [commodity management] > [publish the merchandise] をクリックし、API サービ スとしてアクセスタイプをクリックします。
- 3. 一覧表示したい API グループを選択します (1 つのグループは 1 つの API 商品に対応します)。
- 4. 商品情報を設定して、監査を送信します。

製品が Alibaba Cloud API Marketplace に正常に公開されると、世界中のユーザーがその製品を購入することができるようになります。

2.5 API の削除

API を削除するには、[API Service list] の操作列で、[More] > [Delete] をクリックします。



- APIは、オフラインステータスのときにのみ削除できます。オンラインの場合は、APIを非 推奨にしてから削除します。
- ・ 削除操作は取り消しができません。 API の削除は、注意して行ってください。

2.6 API の呼び出し

本ページでは、API が API Gateway でリリースされた後、API を呼び出す方法について説明します。

API Gateway では、API の権限付与および API の呼び出しに使用する SDK を提供します。 ユーザー自身、ユーザーの同僚、または第三者に対して、API を使用するための権限付与を行う ことができます。 API を呼び出したい場合は、以下の操作を行います。



API を呼び出すための3つの要素

API を呼び出すには、次の3つの要素が必要です。

- ・ API: ユーザーが呼び出そうとしている API です。API パラメーターによって明確に定義されています。
- アプリ: API を呼び出すために使用するアイデンティテです。 AppKey と AppSecret は、 ユーザーのアイデンティティを認証するために提供されています。
- ・ API とアプリの権限関係: アプリが API を呼び出す必要がある場合、アプリは呼び出そうとし ている API へのアクセス許可を持っている必要があります 権限付与により、API へのアクセ ス許可が与えられます。

手順

1. API ドキュメントの取得

取得方法は、APIの取得に使用したチャンネルによって異なります。 通常、API ドキュメ ントは、データマーケットから購入した API サービスに分類され、購入する必要はありませ ん。2 つの取得方法が、プロバイダーにより積極的に指定されています。 詳しくは、「get API documentation」をご参照ください。

2. プロジェクトの作成

アプリは、API を呼び出すために使用するアイデンティティです。 各アプリには、AppKey と AppSecret のセットがあり、これらはアカウントとパスワードに相当します。 詳しく は、「アプリケーションの作成」をご参照ください。

3. 権限の取得

権限付与とは、アプリに API を呼び出す許可を与えることを意味します。 API を呼び出すた めには、まず、ユーザーのアプリで権限付与が行われる必要があります。

権限付与の方法は、APIの入手に使用するチャンネルによって異なります。詳しくは、「許可の取得」をご参照ください。

4. API の呼び出し

API の呼び出し方法は、API Gateway コンソールで提供されている多言語呼び出しのサンプ ルを直接使用する方法、もしくは自己コンパイルされた HTTP または HTTPS リクエストを 使用する方法があります。 詳しくは、「APIの呼び出し」をご参照ください。

2.7 よくある質問

· Q: API Gateway は有効にする必要がありますか。

 A: API Gateway は API ホスティングサービスを提供しています。 自分の API を他のユー ザーに公開する予定の場合は、まず、API Gateway サービスを有効にする必要があります。
 • Q: データソースはどこで設定が行えますか。

A: データソースを作成するには、[DataWorks]、 [Data Integration]、[Data Sources] と、順にクリックします。 設定後、Data Service はデータソース情報を自動的に読み取りま す。

· Q: ウィザードで作成される API と、スクリプトで作成される API の違いは何ですか。

A: スクリプトモードは、より強力な機能を提供します。 詳しくは、「スクリプトモードでの *API* の生成」をご参照ください。 Q: Data Service の API グループとはどういったものですか。 API Gateway における API グループと同様ですか。

A: 特定のシナリオにおいて、ある API グループには複数の API が含まれています。 API グ ループ は最小単位です。一言で言えば、2 つは同等です。 Data Service から API Gateway に API グループを公開すると、ゲートウェイは自動的に同じ名前の API グループを作成しま す。

· Q: API グループを適切に設定する方法を教えてください。

A: 通常、ある API グループには、同様の機能を提供する API や、特定の問題を解決する API が含まれています。 たとえば、都市名で天気を照会する API と、緯度と経度で天気を照 会する API は、APIグループ名を "weather query" とした APIグループに入れます。

· Q: API グループは何個まで作成することができますか。

A: Alibaba Cloud のアカウントでは、最大 100 個の API グループを作成することができます。

・ Q: API の応答出力のページネーションは、どのような状況で有効にする必要があるのでしょうか。

A: 既定では、API は最大 500 レコードを出力します。 API の応答出力のページネーション を有効にすると、さらに多くのレコードを出力することができます。 API のリクエストパラ メーターが設定されていない場合、API は大量のレコードを出力することがあります。その場 合、API の応答出力ページネーションは、自動的に有効になります。

· Q: データソースで作成される API は、POST リクエストに対応していますか。

A: 現在、作成される API は GET リクエストにのみ、対応しています。

・ Q: Data Service は HTTP に対応していますか。

A: 現在、Data Service は HTTP に対応していません。 HTTP は以降のバージョンで、対応 となる可能性があります。