MaxCompute

1. **产品简介**
    1. **简介**
        1. MaxCompute主要服务于**批量结构化数据**的**存储和计算**，可以提供海量**数据仓库**的解决方案以及针对大数据的**分析建模服务。**
        2. **应用**：**数据仓库和BI分析、网站的日志分析、电子商务网站的交易分析、用户特征和兴趣挖掘等。**
    2. **组件介绍**
        1. **数据通道** ：Tunnel (仅提供Java接口)
        2. **计算和分析**
            1. SQL
            2. MR （提供Java接口）
            3. Graph: 图计算处理框架。图计算作业使用图进行建模，图由点(Vertex)和边(Edge)组成，点和边包含权值(Value)。通过迭代对图进行编辑、演化，最终求解出结果，典型应用：PageRank，单源最短距离算法 ，K-均值聚类算法 等等。
        3. **SDK**
        4. **安全**
2. **快速开始**
    1. **准备工作**
    2. **使用大数据平台**
    3. **安装配置客户端**
    4. **添加删除用户**
    5. **用户授权及权限查看**
    6. **添加RAM子账户**
    7. **角色创建及授权**
    8. **创建删除表**
    9. **导入导出数据**
    10. **运行SQL**
        1. **MaxCompute SQL 不支持事务、索引及Update/Delete等操作**
        2. **DDL语句**
        3. **Select语句**
            1. **Group by 操作先于 Select，所以只能接受输入表的列或表达式为key。**
            2. **Order by 必须与 limit 联用。**
            3. **Sort by 前面必须加 distribute by。**
            4. **Order by / sort by / distribute by 只能接受select的输出列（别名）为key（没有别名时即使用列名作为key）**
        4. **Insert语句**
            1. 向静态分区插入数据时，分区列不可以出现在select列表中 。
            2. 动态分区插入时，动态分区列必须在select列表中。
            3. 运行insert 语句时，静态分区必须是高级分区（在前面）
            4. insert overwrite table sale_detail_dypart partition (sale_date='2013', region)
            5. select shop_name,customer_id,total_price, region from sale_detail;
        5. **优化实例**
            1. **join语句中where条件的位置：Join 两张表的时候，从表的分区筛选（子筛选）一定要先完成**
                1. `select * from (select * from A where dt=20150301)A join (select * from B where dt=20150301)B on B.id`
                2. `select * from A join (select * from B where dt=20150301)B on B.id=A.id where A.dt=20150301;`
            2. 窗口函数的优化：首先，窗口函数“over”后面要完全相同，相同的分组和排序条件；其次，多个窗口函数在同一层SQL执行。**符合这两个条件的窗口函数会合并为一个Reduce执行。（如下这种SQL）：**
                1. select
                2. rank() over (partition by A order by B desc) as rank,
                3. row_number() over (partition by A order by B desc) as row_num
                4. from MyTable;
    11. 编写UDF
    12. 编写MapReduce
    13. 编写Graph
3. 基本介绍
    1. 基本概念
        1. 项目空间 Project
            1. 表Table 资源Resource 函数Function 实例Instance
        2. 表 Table
        3. 分区 Partition
        4. 数据类型：Bigint String Boolean Double Datetime
        5. **资源 Resource**
            1. **File类型，Table类型，Jar类型，Archive类型**（zip / tgz / tar.gz / tar / jar）
        6. **函数 Function**
            1. 内建函数
            2. **自定义函数 UDF**
                1. **标量值函数 UDF**

    2. **自定义聚合函数 UDAF**
    3. **自定义表值函数 UDTF**
7. **任务 Task**
    1. 任务 Task 是MaxCompute的基本计算单元
    2. **计算型任务**（能被转化为任务执行计划，实例化）：SQL DML 语句，MapReduce
    3. **非计算型任务**（不能被转化为任务执行计划）：SQL DDL 语句
    4. **非任务请求**：项目空间Project, 资源Resource,自定义函数UDF,实例Instance
8. **任务实例 Instance**
    1. 阶段1：**运行Running**
    2. 阶段2：**结束Terminated - 成功Success / 失败Failed / 取消Canceled**
9. **服务连接**
    1. **网络情况**：公网、阿里云经典网络、阿里云VPC服务
    2. **部署情况**：华东1、华东2、华北2可以用VPC，其他不能用VPC
    3. **下载收费**：公网收费，经典网络和VPC不收费。

2. **常用命令**
    1. 常用命令简介
    2. **项目空间操作**
        1. 如果用户在my_project下想要访问另一项目空间my_project2下的表test_src，则需要指定项目空间名：odps @ my_project>select * from my_project2.test_src;
        2. MaxCompute 没有提供创建及删除项目空间的命令。
    3. **表操作**
        1. 查看分区信息：desc table_name partition(pt_spec) 例：desc meta.m_security_users partition (ds='20151010');
        2. Show Partitions: SHOW PARTITIONS <table_name>;
    4. **实例操作**
        1. **Show Instances**: SHOW INSTANCES [FROM startdate TO enddate] [number]
        2. **Status Instances**: STATUS <instance_id>;
        3. **Kill Instances**: KILL <instance_id>;
        4. **Desc Instances:** DESC INSTANCE <instance_id>;
    5. **资源操作**
    6. **函数操作**
        1. 与**资源文件**一样，**同名函数只能注册一次。**
        2. 一般情况下用户自建函数无法覆盖系统内建函数。**只有项目空间的Owner才有权利覆盖内建函数。**如果用户使用了覆盖内建函数的自定义函数，在SQL执行结束后，会在Summary中打印出warning信息。
    7. **Tunnel命令操作**
        1. upload：帮助用户上传数据到 MaxCompute 的表中；
        2. download：帮助用户从 MaxCompute 的表中下载数据；
        3. resume：如果上传数据失败，通过resume命令进行断点续传，**目前仅支持上传数据的续传。**每次上传、下载数据被称为一个session。在resume命令后指定session id完成续传。
        4. show：查看历史运行信息；
        5. purge：清理 session 目录；
        6. help：输出 tunnel 帮助信息；
    8. **其他操作**
        1. **Alias命令：不改代码的情况下，通过某个固定资源名读取不同资源的方法。**
        2. **Set命令：**设置每个map worker、reduce worker、join worker 的内存大小；设置指定任务下所有worker的内存大小（优先级低于以上三个）；修改每个map worker的输入数据量，修改每个reduce阶段、每个join阶段worker数量；修改MaxCompute指定任务的所有阶段的worker的并发度（优先级低于以上三者）。
        3. **Show Flags**: 显示Set设置的参数
        4. **SetProject**: 设置Project属性或显示当前Project属性配置；
        5. 计算估量**Cost SQL**：预估出一条sql的计量信息，包含输入数据的大小，UDF个数以及SQL复杂等级。

4. **批量数据通道 Tunnel SDK**
    1. 主要接口
        1. TableTunnel **访问 MaxCompute Tunnel服务的入口类**
        2. TableTunnel.UploadSession **上传数据的会话**
        3. TableTunnel.DownloadSession **下载数据的会话**
    2. **TableTunnel**
        1. 生命周期: 从TableTunnel实例被创建开始，一直到程序结束。
        2. 提供创建Upload对象和Download对象的方法提供创建Upload对象和Download对象的方法
    3. **UploadSession**
        1. **生命周期**：从创建Upload实例到结束上传
        2. **创建Upload实例**，可以通过调用构造方法创建，也可以通过TableTunnel创建；请求方式：**同步**
        3. **上传数据**：调用**openRecordWriter**方法，生成RecordWriter实例，其中参数blockId用于标识此次上传的数据，也描述了数据在整个表中的位置，取值范围：[0,20000]，当数据上传失败，可以根据blockId重新上传。请求方式：**异步**
        4. **查看上传**: 调用**getStatus**可以获取当前Upload状态.
           调用**getBlockList**可以获取成功上传的blockid list，可以和上传的blockid list对比，对失败的blockId重新上传。请求方式：**同步**
        5. **结束上传**: 调用**commit(Long[] blocks)**方法，参数blocks列表表示已经成功上传的block列表，server端会对该列表进行验证. 该功能是加强对数据

正确性的校验，如果提供的block列表与server端存在的block列表不一致抛出异常.
Commit失败可以进行重试. 请求方式：**同步**

6. **同一个UploadSession里的blockId (worker)不能重复。**
7. **7种状态说明：**
   - UNKNOWN, server端刚创建一个session时设置的初始值
   - NORMAL, 创建upload对象成功
   - CLOSING, 当调用complete方法(结束上传)时，服务端会先把状态置为CLOSING。
   - CLOSED, 完成结束上传(即把数据移动到结果表所在目录)后
   - EXPIRED, 上传超时
   - CRITICAL, 服务出错

1. **DownloadSession**
   1. **生命周期：**从创建Download实例到下载结束
   2. **创建Download实例**，可以通过调用构造方法创建，也可以通过TableTunnel创建；
      - Server端会为该Download创建一个session，生成唯一downloadId标识该Download，客户端可以通过getId获取
      - 该操作开销较大，server端会对数据文件创建索引，当文件数很多时，该时间会比较长；
      - 同时server端会返回总Record数，可以根据总Record数启动多个并发同时下载
      - 请求方式：**同步**
   3. **下载数据**
      1. 调用**openRecordReader**方法，生成RecordReader实例，其中参数start标识本次下载的record的起始位置，从0开始，取值范围是 >= 0, count标识本次下载的记录数，取值范围是>0。
      2. 请求方式：**异步**
   4. **查看下载：**
      1. 调用**getStatus**可以获取当前Download状态
      2. 请求方式：**同步**
   5. **4种状态说明**
      - UNKNOWN, server端刚创建一个session时设置的初始值
      - NORMAL, 创建Download对象成功
      - CLOSED, 下载结束后
      - EXPIRED, 下载超时

1. **DataHub服务**
   1. **产品简介**
      - 使用**RESTful接口**向用户提供**实时数据的发布(Publish)和订阅**(Subscribe)的功能.
      - **用户将需要上传的数据放入pack中，并指定将这个pack中的数据通过某一路通道(Shard)上传至DHS。**
        1. **同一个pack中的记录必须属于同一个表分区**
        2. **同一个Shard下数据按照上传时间严格有序，且有可能会包含不同Partition的数据**
2. **SQL**
   1. **SQL概要**
      1. **类型 的 转化规则**
         - datetime和string可以互相转换
         - 其他类型和datetime没有互相转换的关系
         - boolean和任何类型都没有互相转换的关系
         - 除datetime和boolean之外，其他类型皆可互相转换
         - 显式和隐式相同。
      2. **=, <>, <, <=, >, >= 的 隐式转化规则**
         1. 有一方是decimal时，皆转化为decimal
         2. 有一方是datetime时，皆转化为datetime（datetime只能和string比较）
         3. 其他情况下都转化为double。
      3. **LIKE, RLIKE, IN 的 隐式转化规则**
         1. source like pattern; source rlike pattern;
            - LIKE和RLIKE 仅接受 string
         2. **key in (value1, value2, ...) 的 隐式转化规则**
            - In右侧的value值列表中的数据类型必须一致；
            - 当key与values之间比较时
              - bigint, double, string，统一转double
              - datetime和string，统一转datetime
              - 除此之外不允许其它类型之间的转换。
      4. **算数运算符 +, -, * , /, %, +, - 的 隐式转化规则**
         - 只有 String、Bigint、Double和Decimal才能参与算术运算, 都转换到double
      5. **逻辑运算符 and, or和not 的 隐式转化规则**
         - 只有boolean才能参与逻辑运算。
      6. **CASE WHEN 的 隐式转化规则**
         - 返回类型只有bigint,double，统一转double
         - 返回类型中有string类型，统一转string（不能转则报错，如boolean类型)
         - 不允许其它类型之间的转换
      7. **String类型与Datetime类型之间的转换**

- 格式为 **yyyy-mm-dd hh:mi:ss**（**0不可忽略；”dd”部分的阈值上限取决于月份实际拥有的天数**）
- 异常
  - cast("2013/12/31 02/34/34" as datetime)
  - cast("20131231023434" as datetime)
  - cast("2013-12-31 2:34:34" as datetime)
  - cast("2013-02-29 12:12:12" as datetime)
  - cast("2013-11-31 12:12:12" as datetime)
- MaxCompute 提供了to_date函数，用以将不满足日期格式的string类型数据转换为datetime类型

2. **运算符**
   1. **关系操作符**
      - **A like B： B是要匹配的模式。 % 匹配任意多个字符； _ 匹配单个字符**
      - **A rlike B： B是正则表达式,不能是空串**
      - **A in B: B 必须是一个常数的集合，至少一项，类型要一致**
   2. **算术操作符**
      - **A/B： AB都为bigint时，返回Double**
      - **只有string, bigint, double可以参与算术运算**
   3. **位操作符**
      1. **A & B ：与**
      2. **A | B ： 或**
      3. **只接受 Bigint**
   4. **逻辑操作符 - True Null False**
      1. **A and B**
         - **FALSE and NULL = FALSE**
         - **TRUE and NULL = NULL**
      2. **A or B**
         - **FALSE or NULL = NULL**
         - **TRUE or NULL =TRUE**
      3. **Not A**
         - **如果A是NULL，返回NULL**
      4. **只接受 Boolean**
3. **DDL语句**
   1. **表操作**
      1. **创建表**
         1. **CREATE TABLE [IF NOT EXISTS]** table_name
            [(col_name data_type [COMMENT col_comment], ...)**]**
            [COMMENT table_comment]
            [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)**]**
            [LIFECYCLE days**]**
            [AS select_statement]

            CREATE TABLE [IF NOT EXISTS] table_name
            LIKE existing_table_name
         2. create table **like**语句
            - **复制表结构（包括分区）**
            - **不复制数据**
         3. create table ... **as select from** ...
            - **复制数据**
            - **不复制表分区结构（分区作为一般列处理）和注释**
         4. like 和 as 都不会继承 Lifecycle
      2. 删除表
      3. **重命名表**
         - **ALTER TABLE** table_name **RENAME TO** new_table_name;
      4. 修改表的**注释**
         - **ALTER TABLE** table_name **SET COMMENT** 'tbl comment';
      5. 修改表的**生命周期**属性
         - **ALTER TABLE** table_name **SET lifecycle** days;
         - **分区表的最后一个分区被回收后，该表不会被删除！！！**
      6. 修改表的**修改时间**
         1. **ALTER TABLE** table_name **TOUCH;**
         2. **生命周期的计算会重新开始**
      7. **清空数据**
         - **非分区表： TRUNCATE TABLE** table_name;
         - **分区表 ： ALTER TABLE** table_name **DROP PARTITION**
   2. **视图操作**
      1. **创建视图**

1. **CREATE** [OR **REPLACE**] **VIEW** [IF NOT EXISTS] view_name
   [(**col_name** [COMMENT col_comment], ...)]
   [COMMENT view_comment]
   [**AS select_statement**]
   *必须有引用表的**读权限**
   *只能**包含一个有效的select语句**
   ***可以引用其它视图，但不能引用自己，也不能循环引用。**
   ***不允许向视图写入数据，例如使用insert into 或者insert overwrite操作视图。**
   - 当视图建好以后，如果视图的引用表发生了变更，有可能导致**视图无法访问，例如：删除被引用表。**用户需要自己维护引用表及视图之间的对应关系。
   - 如果没有指定if not exists，在视图已经存在时用create view会导致异常。这种情况可以用create or replace view来重建视图，**重建后视图本身的权限保持不变。**
2. **删除视图** **DROP VIEW [IF EXISTS]** view_name;
3. **重命名视图** **ALTER VIEW** view_name **RENAME TO** new_view_name;

3. **分区/列操作**
   1. **添加分区**
      - **ALTER TABLE** TABLE_NAME **ADD [IF NOT EXISTS] PARTITION** partition_spec
      - **对于多级分区的表，如果想添加新的分区，必须指明全部的分区值。**
        - **alter table sale_detail add if not exists partition** (sale_date='201312', region='shanghai'); **成功**
        - **alter table sale_detail add if not exists partition**(sale_date='20111011'); **仅指定一个分区sale_date，出错返回**
   2. **删除分区**
      - **ALTER TABLE** TABLE_NAME **DROP [IF EXISTS] PARTITION** partition_spec;
   3. **添加列**
      - **ALTER TABLE** table_name **ADD COLUMNS** (col_name1 type1, col_name2 type2...)
   4. **修改列名**
      - **ALTER TABLE** table_name **CHANGE COLUMN** old_col_name **RENAME TO** new_col_name;
   5. **修改列、分区注释**
      1. **ALTER TABLE** table_name **CHANGE COLUMN** col_name **COMMENT** comment_string;
   6. **同时修改列名及列注释**
      1. **ALTER TABLE** table_name **CHANGE COLUMN** old_col_name new_col_name column_type **COMMENT** column_comment;
   7. **修改表、分区的修改时间**
      1. **ALTER TABLE** table_name **TOUCH PARTITION**(partition_col='partition_col_value', ...);
   8. **修改分区值**
      1. **ALTER TABLE** table_name **PARTITION** (partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...) **RENAME TO PARTITION** (partition_col1 = partition_col_newvalue1, partition_col2 = partiton_col_newvalue2, ...);
      2. **不支持修改分区列列名,只能修改分区列对应的值。**
      3. **修改多级分区的一个或者多个分区值，多级分区的每一级的分区值都必须写上。**

4. **DML语句**
   1. **更新表中的数据(INSERT OVERWRITE/INTO)**
      - **INSERT OVERWRITE | INTO TABLE** tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement **FROM** from_statement
      - **insert into ：追加**
      - **insert overwrite : 更新**
      - **select子句中列的顺序 很重要，插入使需要在顺序上一一对应。**
   2. **多路输出(MULTI INSERT)**
      - **同一分区不能出现多次**
      - **不同分区不能同时有 overwrite 和 into 操作**
   3. **输出到动态分区(DYNAMIC PARTITION)**
      - **如果目标表有多级分区，在运行insert语句时允许指定部分分区为静态，但是静态分区必须是高级分区；**
   4. **SELECT操作**
      1. **distinct 跟着select 走**
      2. **group by 先于 select （因为要改变行数），所以只能接受输入表的列或表达式为Key**
      3. **order by : 全局排序，所以必须使用 Limit**
      4. **distribute by: hash分片**
      5. **sort by: 分片内排序，必须和distribute by联用**
      6. **distribute by/sort by 不能和order by、group by共用**
   5. **子查询**
      - **子查询必须有别名**
   6. **UNION ALL**
      1. **MaxCompute SQL不支持顶级的两个查询结果合并，要改写为一个子查询的形式**
      2. **select * from** (select_statement **UNION ALL** select_statement) **t**;
      3. 对应的**各个子查询的列个数、名称和类型必须一致**；列名不一致时，可以使用 **列的别名** 加以解决。
   7. **JOIN操作**
      - **表名 别名 列名**
      - shop a shop_name

- sale_detail b shop_name
- _____

1. **select** a.shop_name **as ashop**, b.shop_name **as bshop from** shop a **right / left / full outer join** sale_detail b on a.shop_name = b.shop_name ;
2. **select** a.shop_name **from** shop a **(inner) join** sale_detail b on a.shop_name = b.shop_name ;
3. **不支持不等值连接！！！**

8. **MAPJOIN HINT**
   1. **一个大表和一个或多个小表做Join时可以试用mapjoin提升性能**
   2. **大表必须是主表**
   3. **多个表join时，最左边的两个表不能同时是mapjoin的表**
   4. 支持 **不等值** 和 **or逻辑** 等复杂的join条件
9. **HAVING子句**
   1. **WHERE关键字无法与合计函数一起使用，可以采用having字句。**
   2. SELECT Customer,SUM(OrderPrice) FROM Orders
      GROUP BY Customer
      HAVING SUM(OrderPrice)<2000

5. **内建函数-上**
   1. **数学函数**
      1. **ABS** 绝对值, **ACOS** 反余弦, **ASIN** 反正弦, **ATAN** 反正切, **COS** 余弦, **COSH** 双曲余弦, **SIN** 正弦, **SINH** 双曲正弦, **COT** 余切, **TAN** 正切, **TANH** 双曲正切
      2. **CEIL** 天花板 int (>=x)， **FLOOR** 地板 (<=x)
      3. **CONV** 进制转换 conv('1100', 2, 10) = '12'
      4. **EXP** 指数, **LN** 自然对数, **LOG** 以base为底的x的对数： `log(base,x)`, **POW** power: pow(x, y) = x^y, **SQRT** 平方根
      5. **RAND** 随机数： rand(seed)
      6. **ROUND** 四舍五入： `round(123.345, -4) = 0.0`， **TRUNC** 截取到指定小数点位置：trunc(125.815, 2) = 125.81
      7. **SIGN** 符号（判断正负），正返回1，负返回-1， 0返回0
   2. **字符串函数**
      1. **CHAR_MATCHCOUNT** 计算str1中有多少个字符出现在str2中 char_matchcount('abd', 'aabc') = 2
      2. **INSTR** 计算一个子串str2在字符串str1中的位置
      3. **SUBSTR** 返回字符串str从start_position开始往后数，长度为length的子串
      4. **CHR** 将给定ASCII码ascii转换成字符。
      5. **TO_CHAR** 将Boolean、bigint或者double转为对应的string
      6. **IS_ENCODING** 判断字符集转换 is_encoding('测试', 'utf-8', 'gbk') = true is_encoding('測試', 'utf-8', 'gb2312') = false
      7. **MD5** 计算输入字符串value的md5值
      8. **CONCAT** 将参数中的所有字符串连接在一起
      9. **SPLIT_PART** 照分隔符separator拆分字符串str，返回从第start部分到第end部分的子串(闭区间)。split_part('a,b,c,d', ',', 1, 2) = 'a,b'
      10. **TOLOWER** 输出英文字符串source对应的小写字符串。
      11. **TOUPPER** 输出英文字符source串对应的大写字符串。
      12. **TRIM** 将输入字符串str去除左右空格。
      13. **LTRIM** 将输入的字符串str去除左边空格。
      14. **RTRIM** 将输入的字符串str去除右边空格。
      15. **GET_JSON_OBJECT** 在一个标准json字符串中，按照path抽取指定的字符串.
      16. **KEYVALUE** 拆字段，找字典
      17. **LENGTH** 返回字符串str的长度
      18. **LENGTHB** 返回字符串str的以字节为单位的长度。
      19. **REGEXP_EXTRACT** 将字符串source按照pattern正则表达式的规则拆分，返回第occurrence个group的字符。
      20. **REGEXP_INSTR** 返回字符串source从start_position开始，和pattern第n次(nth_occurrence)匹配的子串的起始/结束位置。start_position 默认1 ，nth_occurrence 默认1， return_option 0或1
      21. **REGEXP_REPLACE** 将source字符串中第occurrence次匹配pattern的子串替换成指定字符串replace_string后返回。
      22. **REGEXP_SUBSTR** 从start_position位置开始，source中第nth_occurrence次匹配指定模式pattern的子串。
      23. **REGEXP_COUNT** 计算source中从start_position开始，匹配指定模式pattern的子串的次数。
      24. **REVERSE** 返回倒序字符串
      25. **SPACE** 返回长度为n的字符串
      26. **REPEAT** 返回重复n次后的str字符串
      27. **ASCII** 返回字符串str第一个字符的ascii码
   3. **日期函数**
      1. **DATEADD：按照指定的单位datepart和幅度delta修改date的值**
         错误：select dateadd(2005-03-30 00:00:00, -1, 'mm') from tbl1; 正确： select dateadd(**cast**("2005-03-30 00:00:00" **as datetime**), -1, 'mm') from tbl1;
      2. **DATEDIFF** 计算两个时间date1，date2在指定时间单位datepart的差值。
      3. **DATEPART** 提取日期date中指定的时间单位datepart的值。
      4. **DATETRUNC** 返回日期date被截取指定时间单位datepart后的日期值。
      5. **FROM_UNIXTIME** 将数字型的unix时间日期值unixtime转为日期值。 from_unixtime(123456789) = 2009-01-20 21:06:29
      6. **GETDATE** datetime getdate() 获取当前系统时间
      7. **ISDATE** 判断一个日期字符串能否根据对应的格式串转换为一个日期值，如果转换成功返回TRUE，否则返回FALSE。

8. **LASTDAY** 取date当月的最后一天
9. **TO_DATE** 将一个字符串date按照format指定的格式转成日期值。
10.
11. **TO_CHAR** 将日期类型date按照format指定的格式转成字符串
12. **UNIX_TIMESTAMP** 将日期date转化为整型的unix格式的日期时间值
13. **WEEKDAY** 返回date日期当前周的第几天。
14. **WEEKOFYEAR** 返回日期date位于那一年的第几周，周一作为一周的第一天。某一周大多数日期（4天以上）在哪一年多，算在哪一年

6. **内建函数-下**
   1. **窗口函数**
      1. **COUNT**
      2. **AVG**
      3. **MAX**
      4. **MIN**
      5. **MEDIAN**
      6. **STDDEV**
      7. **STDDEV_SAMP**
      8. **SUM**
      9. **DENSE_RANK 计算连续排名。 col2相同的行数据获得的排名相同。**
      10. **RANK 计算排名。col2相同的行数据获得排名顺序下降。**
      11. **LAG 按偏移量取当前行之前第几行的值**
      12. **LEAD 按偏移量取当前行之后第几行的值**
      13. **PERCENT_RANK 计算一组数据中某行的相对排名。**
      14. **ROW_NUMBER 计算行号，从1开始。**
      15. **CLUSTER_SAMPLE 分组抽样。想要从每组中抽取约10%的值，可以用以下MaxCompute SQL完成：**

```
select key, value
from (
  select key, value, cluster_sample(10, 1) over(partition by key) as flag from tbl
  ) sub
where flag = true;
```

   2. **聚合函数**
      1. **COUNT count(*)** 返回包括Null， count(col_name)不包括Null
      2. **AVG MAX MIN MEDIAN SUM**
      3. **STDDEV STDDEV_SAMP**
      4. **WM_CONCAT** 用指定的spearator做分隔符，链接str中的值。
   3. **其他函数**
      1. **CAST** 转换成目标类型
      2. **COALESCE** 返回列表中第一个非NULL的值，如果列表中所有的值都是NULL则返回NULL
      3. **DECODE** 实现if-then-else分支选择的功能。
      4. **GREATEST LEAST**
      5. **ORDINAL** 将输入变量按从小到大排序后，返回nth指定的位置的值。
      6. **UUID** 返回一个随机ID
      7. **SAMPLE** 对所有读入的column_name的值，sample根据x，y的设置做采样，并过滤掉不满足采样条件的行。
      8. **CASE WHEN表达式**
      9. **IF**
7. **UDF**
   1. **UDF** 用户自定义标量值函数(User Defined Scalar Function) **一对一**
   2. **UDTF** 自定义表值函数 **一对多**
   3. **UDAF** 自定义聚合函数 **多对一**
   4. **用udtf读取ODPS资源的示例**
      1. **编写udtf程序，编译成功后导出jar包(udtfexample1.jar)**
      2. **添加资源到ODPS:**
         - **Add file** file_resource.txt;
         - **Add jar** udtfexample1.jar ;
         - **Add table table_resource1 as** table_resource1 **;**
         - **Add table table_resource2 as** table_resource2**;**
      3. **在ODPS中创建UDTF函数(my_udtf)：**
         - **create** function mp_udtf **as com.aliyun.odps.examples.udf.UDTFResource using** 'udtfexample1.jar, file_resource.txt, table_resource1, table_resource2';
      4. **在odps创建资源表table_resource1、table_resource2, 物理表tmp1。并插入相应的数据。**
      5. **运行该udtf。**
8. **附录**
   1. **转义字符**
   2. **like 匹配 ： %多个 _单个**

3. **MapReduce**
   1. **应用场景**
      1. **搜索**

2. **Web访问日志分析**
3. **文本统计分析**
4. **海量数据挖掘**
5. **机器学习**
6. **自然语言处理**
7. **广告推荐**

2. **处理流程**
   1. **分片，每一片对应一个Map Worker同时处理**
   2. **Map Worker输出时给每一条数据指定一个Key, 这个Key值决定了这条数据将会被发送给哪一个Reduce Worker。相同Key的数据发送给同一个Reduce Worker，一个Reduce Worker可以有多个Key。**
   3. **进入Reduce阶段前，先Shuffle，即让有相同Key的数据彼此相邻。**
3. **由于 MaxCompute 的所有数据都被存放在表中，因此MaxCompute MapReduce的输入、输出只能是表，不允许用户自定义输出格式，不提供类似文件系统的接口。**

4. **图模型**
   1. **概要**
      1. **图由点(Vertex)和边(Edge)组成，点和边包含权值(Value)**
      2. **Graph数据结构**
         - **< ID, Value, Halted, Edges >，分别表示点标识符(ID)，权值(Value)，状态(Halted, 表示是否要停止迭代)，出边集合(Edges，以该点为起始点的所有边列表)**
      3. **Graph 程序逻辑**
         1. **加载图**
         2. **迭代计算**
            1. **一次迭代为一个”超步”(SuperStep)，遍历所有非结束状态(**Halted值为false**)的点或者收到消息的点(**处于结束状态的点收到信息会被自动唤醒**)**，并调用其**compute(ComputeContext context, Iterable messages)方法：**
               处理上一个超步发给当前点的消息*(Messages)*；
               根据需要对图进行编辑：*1).* 修改点/边的取值；*2).* 发送消息给某些点；*3).* 增加/删除点或边；
               通过*Aggregator*汇总信息到全局信息；
               设置当前点状态，结束或非结束状态；
               迭代进行过程中，框架会将消息以异步的方式发送到对应*Worker*并在下一个超步进行处理，用户无需关心；
         3. **迭代终止（满足以下任意一条）：**
            1. **所有点处于结束状态(Halted值为true)且没有新消息产生；**
            2. **达到最大迭代次数；**
            3. **某个Aggregator的terminate方法返回true；**
   2. **图模型功能介绍**
      1. **运行作业**
      2. **输入输出**
      3. **读取资源**
         1. **GRAPH程序中添加资源**
         2. **GRAPH程序中使用资源**
   3. **SDK介绍**
   4. **图模型开发和调试**
   5. **应用限制**
   6. **Aggerator机制介绍**
1. **流式计算**
2. **计量计费**
   1. **存储计费**
      - **包括表(Table)和资源(Resource)**
      - **计费周期是：天**
   2. **计算计费**
      1. **按量后付费**
         - **一次SQL计算费用 = 计算输入数据量 * SQL复杂度 * SQL价格**
      2. **预留资源计费**
   3. **一次下载费用 = 下载数据量 * 下载价格**
   4. **欠费预警与停机策略**
3. 下载中心
4. 安全指南
5. SDK
6. 工具
7. 产品使用问题

DataIDE

1. 产品简介
   1. **产品概述**
      1. **产品架构**：基于MaxCompute（原ODPS）的集成开发环境，包括数据开发、数据管理、数据分析、数据挖掘和管理控制台。
      2. **主要特性**

1. 拖拽式的操作界面
2. 个性化数据收藏与管理
3. 一键式跨项目任务发布
4. 可视化任务监控

3. **开发流程**
    1. 数据产生、数据收集与存储、分析与计算、数据提取、数据展现与分享
4. **产品优势**
    1. 大数据处理能力
    2. 权限管理机制
    3. 易用
    4. 数据管理机制

2. **使用场景**
    1. 将业务系统产生的数据轻松上云，构建大型数据仓库和BI应用，利用ODPS强大的海量存储与数据处理能力。
    2. 基于数据开发快速使用和分析数据，将大数据加工结果导出后直接应用于业务系统，实现数据化运营。
    3. 针对作业调度与运维复杂性，数据开发提供统一友好的调度系统和可视化调度运维界面，解决运维管理不便等问题。

2. 快速开始
    1. 开通数据开发套件
    2. 创建项目空间
    3. 添加成员及授权
    4. 创建删除表
    5. **数据导入说明**
        1. **对于数据库类型** 采用DataIDE数据同步
        2. **本地文件导入** 文件<10M
        3. **使用dataX** 大批量本地文件导入
    6. 本地文件上传
    7. **创建任务**
        1. 工作流任务 or 节点任务
        2. 一次性调度 or 周期性调度
    8. **创建数据同步任务**
        1. **创建数据表**
        2. **新建数据源**
        3. **新建任务**
        4. **配置数据同步任务（任务的节点组件里选数据同步）**
    9. **创建UDF**
        1. **代码编写**
        2. **添加jar资源（上传资源）**
        3. **注册UDF函数（新建函数）**
        4. **在ODPS SQL中试用函数**
    10. **创建OPEN MR**
        1. **开通OPEN MR(申请使用MR/Shell)**
        2. **数据表准备**
        3. **编写MapReduce程序**
        4. **添加资源**
        5. **创建OpenMR节点(任务的节点组件里可以选Open MR)**
        6. **查看结果**
    11. 创建Shell任务
    12. **测试任务**
        1. 点击顶部运维中心，导航至任务管理>**任务管理**视图。
        2. 选择buy_seller_interactions_feature工作流，**右键测试任务。**
        3. 在测试运行弹出框中**选择业务日期**，点击生成并运行。
        4. 点击**前往查看冒烟结果。**
        5. 跳转至**任务运维**视图>**测试**tab页，查看工作流运行情况。
    13. **下线工作流：数据开发-任务开发-右键删除工作流**

3. **数据开发手册**
    1. 功能概述
    2. 权限点划分
    3. 文件目录
    4. 工作流设计器
        1. **依赖 & 跨周期依赖**
            1. 依赖属性配置的调度依赖是**同周期依赖和跨周期依赖不冲突。**任务A可以配置依赖属性依赖任务B也可以配置跨周期依赖依赖B，如此任务A即依赖任务B本周期也依赖任务B上周期。
            2. 若**任务A是小时/分钟调度**，**任务B为天调度**，任务B配置依赖任务A的上周期，那么当天B任务的实例会**依赖A任务昨天所有实例。**
            3. 若**任务A和B都是小时/任务调度**，调度周期一样，任务B配置依赖任务A的上周期，则任务B每个实例都将**依赖A昨天所有实例和A与B同周期的前一个周期实例。**
            4. 如天任务A配置跨周期依赖B的上周期，那么对A任务进行补数据的时候，补数据实例会去依赖B任务自动调度上周期实例，如果自动调度的

上周期实例不存在则不依赖。

5. ODPS代码编辑器
6. **系统调度参数**
    1. **系统参数**
        1. **${bdp.system.bizdate}** yyyymmdd **日常调度实例定时时间的前一天**（年月日）
        2. **${bdp.system.cyctime}** yyyymmddhh24miss **日常调度实例定时时间（年月日时分秒）**
    2. **自定义参数**
        1. **声明变量方式：${变量名}**
        2. **基于系统参数${bdp.system.cyctime}进行自定义格式**
7. 发布管理

4. 数据管理手册
    1. 基本介绍
    2. 功能权限点划分
    3. **全局视图**
    4. **查找数据**
    5. 表详情页介绍
    6. 数据权限申请
    7. **数据表管理**
    8. 创建表
    9. 权限管理
    10. **管理配置**
5. 运维中心手册
    1. 基本介绍
        1. 概览
        2. **任务管理：任务管理视图、任务管理列表**
        3. **任务运维：任务运维视图、任务运维列表**
        4. 监控报警
    2. **权限划分**
        1. **开发人员**
            1. 可操作单个工作流/节点测试、补数据、暂停、重跑任务，查看任务运行日志等操作
            2. 还可配置监控报警。
        2. **运维人员**
            1. 单个工作流/节点测试、补数据、暂停、重跑任务等操作
            2. 批量修改工作流/节点属性、批量杀任务及批量重跑、配置监控报警等干预性的操作；
        3. **项目管理员**
            1. 同运维
    3. 概览
    4. 任务管理
    5. 监控报警
    6. 任务运维
        1. **节点实例右键操作特别说明：**
            1. **终止** 只能终止等待时间、等待资源、运行中状态的任务。
            2. **重跑并恢复调度** 只能重跑未运行、成功、失败状态的任务。
            3. **置成功并恢复调度** 只能失败状态的任务能被置成功。
            4. **重跑下游并恢复调度** 只能勾选未运行、完成、失败状态的任务，如果勾选了其他状态的任务，页面会提示“已选节点中包含不符合运行条件的节点”，并禁止提交运行。
            5. **暂停：暂停节点当前周期的调度。** 此操作只暂停当前周期，下一周期仍会正常调度，如果要永久暂停调度，需要到流程设计器中，找到“节点>调度>时间属性>节点状态>”，勾选”暂停调度”，再保存、发布方可生效。
            6. 恢复：相对上面的“暂停“操作。
        2. **补数据**
6. **组织管理手册**
    1. **项目管理**
        1. 编辑项目，可编辑项目描述，添加项目管理员绑定调度资源
        2.
    2. **调度资源**
        1. **新建调度资源**
            1. **应用实例**
                1. 大量任务处于“等待资源状态”；当前项目的任务量达到一定的量值，已有gateway的资源已经不满足业务需求，需要扩大资源，申请添加gateway。【注意】：**数据同步任务**一个任务会占用10个机器槽位（即10个并发数），默认资源组资源有限，所以在做数据生产过程中，**数据同步任务**一般都需要用自建资源组来运行以保证任务不会一直等待资源。
                2. 项目下有一些特殊的任务（**比如某些shell脚本**）需要在特定的机器上执行，需申请自己的gateway作为任务执行机器。
        2. **配置服务器**
            1. 专用调度资源绑定服务器不超过1000台，且只能从属于该组织范畴内的ECS机器列表中选取，同时一台ECS虚机只能属于一个自定义调度资源。
            2. **实现过程**
                1. **步骤1**：购买ECS云服务器。

2. **步骤2**：查看ECS主机名和IP地址。
3. **步骤3**：组织管理员进入数据开发套件，导航至**组织管理>调度资源**，点击增加调度资源，将购买的ECS云服务器添加到资源组。
4. 步骤4：经过上述步骤后，已经将新购买的ECS信息注册到了数据开发套件中，但是目前为止还不能服务。如果是新添加机器，请按照如下步骤操作：**执行初始化**

7. 项目管理手册
    1. **项目属性**
        1. **基本属性配置**
            1. 本配置项仅“**项目管理员**”可以操作，其他项目成员只可查看。
            2. **发布目标**场景：一个组织内创建了2个项目用于完成一个业务需求，项目A只做工作流开发过程，项目B只做日常生产调度,工作流都由项目A发布过来，那么可以配置项目A的发布目标设置成项目B。
        2. **数据源配置**
        3. **计算引擎配置**
        4. **流程控制**
    2. **成员管理**
        1. **组织管理员**
            1. 指组织的管理者，可新建计算引擎、新建项目空间、新建调度资源、添加组织成员、为组织成员赋予**组织管理员角色**、配置数据类目等。
        2. **项目管理员**
            1. 指项目空间的管理者，可针对项目空间基本属性、数据源、当前项目空间计算引擎配置和项目成员进行管理，并为组织成员赋予**项目管理员**、开发、运维、部署、访客角色。
        3. **开发**
            1. 开发角色用户能够创建工作流、脚本文件、资源和UDF，新建表，同时**可以创建发布包，但不能执行发布操作。**
        4. **运维**
            1. 项目空间的运维人员，由项目管理员/项目所有者分配运维权限；**拥有发布及线上运维的操作权限**，无数据开发的操作权限
        5. **部署**
            1. 部署角色与运维角色相似，但是其**没有线上运维操作权限。**
        6. **访客**
            1. 访客角色的用户只具备查看权限，而无权限进行编辑工作流和代码等。
8. 基础示例
9. 进阶示例

数据集成

1. **优势**
    1. **多** **多样的数据通道支持；齐全的数据传输方式；丰富的数据处理插件**
    2. **快** **高效的调用方式；强劲的传输速度；强大的吞吐能力**
    3. **好** **健壮的传输通道；智能的错误检测；自动的传输恢复**
    4. **省** **无需构建、即入即用；动态分配、弹性伸缩；按需申请、按量付费**
2. **功能**
    1. **批量数据同步**
    2. **定时任务调度**
    3. **任务出错报警**
3. **应用场景**
    1. **云产品之间**的数据同步
    2. **定时同步**业务**增量**数据
    3. **本地机房**存量数据入云
4. 产品简介
5. **快速开始**
    - **定时运行**
        - cron表达式
            - **00 00 00 * * ?**
            - **秒 分 小时 天 月 周**
            - **天和周不能同时设置为 * ， ？表示忽略**
    - **想每次运行时，抽取不同时间的数据**
        - **yyyyMMdd HH:mm:ss +/- N （y m d h mi）**

CDP利用**RDSReader(实际上是MysqlReader)完成数据从RDS抽取，并转换为CDP的数据协议**，投递给CDP传输中间层。
CDP利用**ODPSWriter完成CDP的数据协议转换为ODPS的数据协议**，并将转换后的数据向**ODPS写**入。

- **单个传输单元数据吞吐带宽为1MB/s，会上下波动**
- 对于字段的映射是按照配置**字段顺序**进行一一映射
    - 源端字段和目标端**字段个数一致** 可以
    - **源端字段比目标端字段多 报错**
    - **源端字段比目标端字段少 添加null**

- **幂等性是通过Writer插件的前置条件来实现的，**例如用户可写ODPS配置中提供数据写入前的清理工作，该配置可以保证每次数据导入前都会先清除当前表或者分区的现有数据，这样能够保证数据多次写入的结果和当前一次性写入结果一致。
- 唯一性保证
- 原子语义：建两个表
- 增量同步
  - 时间戳方式：自动&手动
  - 触发器：触发后写入一个临时表，然后传输临时表的数据即可
  - 日志表方式：读日志表
- 脏数据

其他产品

RDS

- **可弹性伸缩**
- **在线数据库服务**
- **全SSD盘高性能存储**

应用场景

- 数据异地容灾
- 读写分离
- 多结构数据存储-结构化的放到RDS
- 搜索引擎——复杂关键词，亿级别百毫秒

DRDS 分布式关系型数据库服务

无限扩容，弹性扩展

OTS

移动社交：

- 波动明显
- 大并发
- 低延时

金融风控

电商物流

日志监控

ADS

实时 高并发 在线分析

传统企业BI引擎

新型BI

OSS

方便、灵活的使用方式，让您像操作本地文件一样

- 提供标准的RESTful API接口，丰富的SDK包，客户端工具、控制台，您可以像使用文件一样，方便上传/下载、检索、管理用于Web网站或者移动应用海量的数据。 了解详情>>
- 支持流式写入和读出，特别适合视频等大文件的边写边读业务场景。 了解追加上传方式详情>>
- 支持数据生命周期管理，您可以方便自定义到期数据转入低成本的归档服务或者批量删除。 了解详情>>

流式写入和读取，支持边读边写

- 尤其适用视频监控存储、提供类文件读写接口，使摄像机的视频码流能像“流”一样往对象（文件）后面追加新内容，且已上传的内容即使在该对象（文件）未写完时都可以被访问，真正实现文件流式存储，视频录像秒级回放 了解详情>>
- 提供设备端直连OSS的C语言SDK开发包和Demo。加快您的开发速度，节省开发成本 下载解决方案SDK>>

来源： https://www.aliyun.com/product/oss?spm=5176.8142029.388261.45.5lUj6P