Perforce 2007.1
P4Report User's Guide

March 2007

Table of Contents

Preface

About This Manual

This manual describes the Perforce Reporting System (P4Report), which enables you to create reports by accessing the database where Perforce stores information about the files, users, clients, and jobs in its control. To access the database, you use SQL with an ODBC driver. Microsoft defines ODBC as follows:

> "Open Database Connectivity (ODBC) is a widely accepted application programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language."

(From the Microsoft web page `http://www.microsoft.com/data/odbc/`)

After you install P4Report, you can use tools such as Microsoft Excel or Crystal Reports (or any tool that can interact with an ODBC data source) to report on Perforce activity. This guide provides examples of typical queries and reports.

P4Report includes the Perforce SQL Command-Line Client (P4SQL), which enables you to test queries against your Perforce server before embedding the queries into a reporting tool. To use the tool, choose **Start>Perforce SQL>SQL Command Line Client**. P4Report launches P4SQL in a DOS window. At the SQL prompt, you can issue valid SQL-92 queries. For details about using the command line client, refer to "Using the Perforce SQL Command-Line Client" on page 11.

For details about the SQL functions supported by P4Report, refer to Appendix B, P4Report Functions. For details about the schema of the Perforce database, refer to Appendix A, Database Schema and SQL Keywords.

Please Give Us Feedback

Does this guide provide the information you need to succeed with the product? Email comments and questions to `manual@perforce.com`.

Chapter 1 Configuring and Using P4Report

To install P4Report, download the installer from the Perforce web site and run it. The installer enables you to specify where you want P4Report installed and to choose from available options. When you install P4Report, the installation program configures a default ODBC data source using the Perforce settings in effect on your client computer. To access other Perforce servers, you can modify the default data source or configure additional ODBC data sources, one for each server. The following section tells you how.

Configuring an ODBC Data Source

To configure an ODBC data source for a specific Perforce server, perform the following steps:

1. Choose **Start>Perforce>P4Report>Data Sources**.

   The **ODBC Data Source Administrator** dialog is displayed.

2. Click **Add...** The **Create New Data Source** dialog is displayed.

3. Scroll to **Perforce ODBC Driver**, click it, then click Finish. The **ODBC P4Report Setup** dialog is displayed.

4. Enter the required information as follows:

   - **Data Source Name:** assign a name that corresponds to the Perforce server from which you intend to retrieve data. (The **Description** field is optional.)
   - **Port**: the host computer and TCP/IP port number on which the Perforce server listens for client requests. Corresponds to the Perforce `P4PORT` setting.
   - **User**: the name of the Perforce user that P4Report uses to connect to the Perforce server. Corresponds to the Perforce `P4USER` setting.
   - **Client**: the name of the Perforce client specification that P4Report uses to connect to the Perforce server. Corresponds to the Perforce `P4CLIENT` setting.
   - **Host**: the name of the Perforce client computer. Corresponds to the Perforce `P4HOST` setting.

     To configure the data source to use the Perforce settings in effect on the client computer at run time (instead of specifying fixed settings), check the **Default** boxes for the desired settings. For details about Perforce settings, refer to *Introducing Perforce*.

To configure the data source so that ODBC client programs can update the underlying data, check the **Enable Updates** box.

5. Click **OK**. Your data source is added to the list of data sources displayed on the **ODBC Data Source Administrator** dialog.

Reporting Using Microsoft Excel

To create reports based on Perforce data, you can use any reporting tool that accepts data from an ODBC data source. This guide tells you how to create reports using Microsoft Excel.

Before you define queries in Excel, be sure to define all required data sources as described in the preceding section, "Configuring an ODBC Data Source" on page 9.

To enter the SQL queries in the guide into Microsoft Excel spreadsheets, you must have Microsoft Query installed. There are two ways you can query the Perforce database:

- Query Wizard: the Query Wizard enables you to define simple queries graphically.
- SQL: For complex queries, such as queries that use SQL functions that are not available in the Query Wizard, you can enter SQL. You can use the SQL queries in this guide as templates.

Creating Queries Using the Query Wizard

To create queries using the Query Wizard, perform the following steps.

1. Launch Excel. A new, empty spreadsheet is displayed.
2. Choose **Data>Get External Data>New Database Query...** The **Choose Data Source** dialog is displayed.
3. On the **Databases** tab, choose the Perforce data source from the list of data sources and click **OK**. The **Query Wizard - Choose Columns** dialog is displayed.
4. Expand the desired tables, choose the desired columns column, and click **>** to move the columns into the **Columns in your query** list.
5. Click **Next**. The **Filter Data** dialog is displayed.
6. Enter any conditions you want to apply to the data, to extract only the data on which you want to report.
7. Click **Next**. The **Sort Data** dialog is displayed.
8. Specify any sort orders you want to apply to the data. For example, to display time or date fields starting with the most recent, sort these fields in descending order.

9. Click **Next**. The **Finish** dialog is displayed. Click **Finish** to return the data to your spreadsheet.

Entering SQL Queries in Microsoft Excel

To enter an SQL query into a spreadsheet, perform the following steps.

1. Launch Excel. A new, empty spreadsheet is displayed.
2. Choose **Data>Get External Data>New Database Query...** The **Choose Data Source** dialog is displayed.
3. On the **Databases** tab, choose the Perforce data source from the list of data sources and click **OK**. The **Query Wizard - Choose Columns** dialog is displayed.
4. Expand any table, choose any one column, and click **>** to move the column into the Columns in your query list. It does not matter which column you choose, because you will replace the entire query with your own.
5. Click **Next**. The **Filter Data** dialog is displayed.
6. Click **Next**. The **Sort Data** dialog is displayed.
7. Click **Next**. The **Finish Dialog** is displayed.
8. Choose the **View or edit data in Microsoft Query** option and click **Finish**. Microsoft Query is launched, displaying your initial query.
9. Choose **View>SQL...** The **SQL** dialog is displayed.
10. Enter your SQL query in the SQL window and click **OK**. If your SQL query is correctly formed and the Perforce database contains any data that fulfills the query conditions, Microsoft Query displays the data in its main window.
11. To place the data in your spreadsheet, choose **File>Return Data to Microsoft Excel**.

Using the Perforce SQL Command-Line Client

P4SQL, the Perforce SQL Command-Line Client, enables you to query Perforce servers directly using SQL, without going through Microsoft Excel, Crystal Reports, or other reporting software. P4SQL runs in a DOS window and accepts SQL-92-compliant queries. By default P4SQL queries the server specified by the Perforce settings in effect on your client computer (`P4PORT`, `P4USER`, `P4CLIENT`, and `P4HOST`) and does not use an ODBC data source. To specify an ODBC data source, use the `-n` flag when you invoke P4SQL.

Launching the command line client

To launch P4SQL, choose **Perforce>P4Report>SQL Command-Line Client**. A DOS window is launched and the command line client is started. Alternately, open a DOS window, change to the directory where you installed P4Report, and run `p4sql` from the command line. When invoked from the command line, the SQL command line client accepts the following flags:

Flag	Description
`-c client`	Set client name (default `P4CLIENT`)
`-d "delimiters"`	Specify the column delimiter characters for output. You can specify a single delimiter character or a string (multiple characters). If the delimiter character has meaning to the command shell, enclose it in double quotes. For example, to put a comma between column values, invoke P4SQL AS follows: `p4sql -d ","`
`-e`	Enable updates and inserts.
`-H host`	Set host name (default `P4HOST`)
`-h or -?`	Print usage message
`-i file`	Execute the SQL queries in the specified text file.
`-n ODBC_data_source`	Specify an ODBC data source to query
`-P password`	Set user's password (default `P4PASSWD`)
`-p port`	Set server port (default `P4PORT`).
`-q`	Set quiet mode (suppresses header/footer output)
`-s query`	Run a single SQL statement. For example: `p4sql -s "select * from jobs"`
`-u user`	Set user's Perforce username (default `P4USER`)
`-V`	Print version information

Running queries

At the `SQL>` prompt, enter your queries (for example, `select * from users;`). By default you cannot alter the Perforce metadata using P4Report. To enable updating, start P4SQL from the command line, specifying the `-e` flag (or add the `-e` flag to the command associated with the **SQL Command Line Client** menu shortcut).

Note that typing RETURN or ENTER does not submit your query. You must terminate every query with a semicolon (;).

You can create text files that contain queries and run them from the command line by invoking the command line client, specifying the `-i` flag. You can send query output to an output file. By default, the query output returned by P4SQL is tab-delimited, enabling you to import it into Microsoft Excel. For example, to store query output in a file named `output.txt`, invoke P4SQL as follows:

```
C:\Program Files\Perforce\p4sql.exe -i query.txt > output.txt
```

The LIKE clause

In P4Report queries, `LIKE` clauses are case-sensitive so they can be sent verbatim to a UNIX Perforce server.

The SQL wildcards _ and `%`, which match single and multiple characters respectively, are translated by P4Report into Perforce wildcards as follows:

SQL wildcard	Perforce wildcard
`%`	`...`
_	*

ODBC syntax vs. SQL-92 syntax: joins and escape sequences

Join syntax

The SQL-92 syntax for expressing joins is as follows (words in square brackets [] are optional):

```
SELECT * FROM table-reference [NATURAL] <join type> JOIN table-reference
```

where `<join type>` is one of the following:

- `INNER`
- `LEFT [OUTER]`
- `RIGHT [OUTER]`
- `FULL [OUTER]`
- `UNION`
- `CROSS`

The ODBC standard defines its own outer join syntax, as follows:

```
SELECT * from {oj table-reference {LEFT | RIGHT | FULL}
OUTER JOIN {table-reference}
```

P4Report supports the ODBC join syntax, but the syntax is primarily designed for ODBC applications that generate queries and require a standard way of specifying outer join queries. For purposes of simplicity, SQL-92 join syntax is recommended.

Escape sequences

Curly brackets ({}) are used by ODBC for escape sequences that contain standard syntaxes for the following SQL features:

- Date, time, timestamp, and datetime interval literals
- Scalar functions such as numeric, string, and data type conversion functions
- `LIKE` predicate escape character
- Procedure calls
- Outer joins

Note that P4Report discards ODBC escape sequences when parsing an SQL expression, thereby converting the expression into a standard SQL one. If you use the preceding features, use standard SQL syntax and do not use ODBC escape sequences.

Query Performance

Avoid uncorrelated subqueries

For best performance, avoid queries that contain uncorrelated subqueries, because the subquery is executed for every row returned by the `SELECT` clause. Instead of subselects, use joins.

For example, instead of this subselect:.

```
select user from users where user in (select owner from clients);
```

use a simple join:.

```
select user from users, clients where user = owner;
```

Memory and CPU usage

Queries that contain an `ORDER BY` or `GROUP BY` clause require P4Report to load the report set into memory for sorting. For such queries, memory use increases in proportion to the number of rows in the result set, the number of columns in each row and the size of each column.

When a query is running, the SQL engine processes data until the query is complete. To process the data as fast as possible, the engine uses all the CPU time that it is given. If you monitor query performance, you are likely to see high CPU consumption during query processing.

Chapter 2 Administrative Reporting

This chapter contains queries that return information about Perforce clients and users, typically of interest to system administrators. You can use these queries in Microsoft Excel, Crystal Reports, and other tools that can be configured with ODBC data sources. For details about executing SQL queries in Excel, see "Reporting Using Microsoft Excel" on page 10. For details about supported functions, refer to Appendix B, P4Report Functions.

What users made the most changes?

```
select user, count(*) from changes
     group by user
     order by 2 desc;
```

What has a specified user done in the past 24 hours?

```
select change, description from changes
     where user='jones' and timestampdiff(3, date, curtime()) < 24;
```

What clients haven't been used in a year or more?

```
select client, owner, accessed from clients
     where timestampdiff(8, accessed, curtimestamp())>0;
```

What users haven't been used in a year or more?

```
select user, fullname, accessed from users
     where timestampdiff(8, accessed,  curtimestamp())>0;
```

What clients belong to a specified user?

```
select client, owner, accessed from clients
     where owner='jones';
```

How many files/which files does a specified user have open?

```
select client, count(*) from opened
     group by client
     order by 2 desc;
```

Which groups contain a specified user?

```
select groupname from groups
     where userspec='tonyz';
```

What users have superuser privileges?

```
select * from protections
     where mode = 'super' and type  = 'user';
```

Note The results of queries against the protections table can be misleading. Identifying paths in the protection table that have a specified access level is not the same as determining the access level for that path. P4Report can report on records in the protections table that match an evaluation condition (user, path, access), but P4Report cannot actually compute the protections. Computing protections requires knowledge of the order of the protection lines, exclusionary mappings, and Perforce wildcards.

Chapter 3 Defect Tracking

Overview

You can change the structure of Perforce jobs by altering the job specification (see the *Perforce System Administrator's Guide* for details). The queries in this chapter are based on a job specification containing the following fields:

```
dependency          release
description         reportedby
job                 reporteddate
modifiedby          severity
modifieddate        status
ownedby             subsystem
priority            type
```

You cannot create a query or report on one server and then execute it against a server which has a different job specification. If you change the job specification, you might need to change the corresponding P4Report queries, especially if you rename or remove fields.

Note that job status can be easily changed, and a job can be reopened after being closed. The examples in this section assume that the defect tracking process does not allow jobs to be reopened.

For details about supported SQL functions, refer to Appendix B, P4Report Functions.

What jobs are unassigned?

```
select job from jobs
     where ownedby = '' and status='open';
```

What jobs were entered by a specified user?

```
select * from jobs
     where reportedby = 'tonyz' and status='open';
```

What jobs of a specified type are open?

```
select * from jobs
     where type = 'bug' and status = 'open';
```

How many jobs were closed in a specified branch this quarter?

```
select count(job) from jobs
     where filespec='//depot/main/release1.1...'
     and status='closed'
     and quarter(modifieddate)=quarter(curdate());
```

What changes are not associated with any fixes?

```
select change, description, fixes.job
     from changes left outer join fixes
     where fixes.job is NULL;
```

A simple Excel pie chart: display open jobs by job type

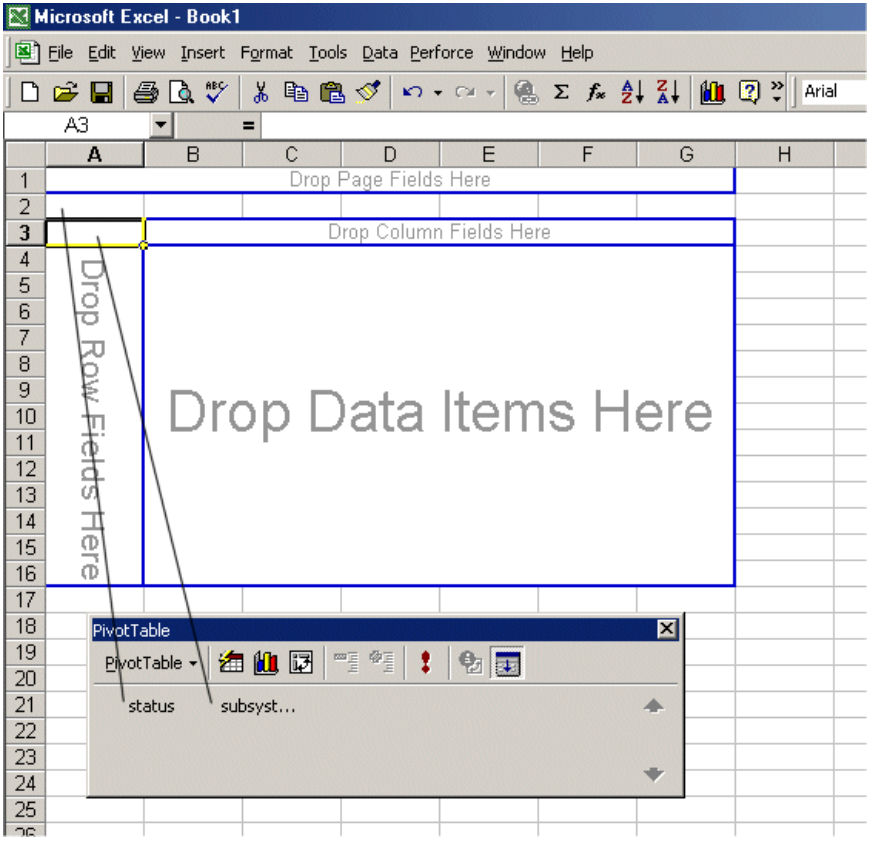
You can create an Excel pie chart that displays the percentage of total open jobs assigned to each component you track using Perforce. Note that the Perforce job specification can be customized. This example assumes that your job specification includes a “status” field and a “subsystem” field.

To load the data, perform the following steps:

1. Launch Excel. An empty spreadsheet is displayed.
2. Choose **Data>Get External Data>New Database Query...** The **Choose Data Source** dialog is displayed.
3. On the **Databases** tab, choose the Perforce data source from the list of data sources and click **OK**. The **Query Wizard - Choose Columns** dialog is displayed.
4. Expand the **jobs** table and move the **status** and **subsystem** columns into the **Columns in your query** list.
5. Click **Next**. The **Filter Data** dialog is displayed.
6. Click the **status** field. In the **Only include rows where** group, choose equals from the left drop-down list of conditions, and enter “open” in the field on the right.
7. Click **Next**. The **Sort Data** dialog is displayed.
8. From the drop-down list, choose “subsystem” and click **Next**. The **Finish** dialog is displayed. Click **Finish** to return the data to your spreadsheet.
9. On your spreadsheet, Excel displays the **Returning External Data to Microsoft Excel** dialog. Specify where you want the data placed and click **OK**. Two columns of data are loaded into your spreadsheet.

To create the chart from the data you loaded, perform the following steps:

1. Select the data by clicking the buttons at the top of the columns. (For example, if you loaded data starting at the top right cell, click the **A** and **B** buttons.)
2. Choose **Data>PivotTable and PivotChart Report**. The PivotTable and PivotChart wizard is launched and displays the **Step 1 of 3** dialog.
3. Select the **Microsoft Excel list or database** and **PivotTable** options and click **Next**. The **Step 2 of 3** dialog is displayed, listing the range of data in the selected columns.
4. Click **Next**. The **Step 3 of 3** dialog is displayed, asking where you want to put the chart.
5. Choose **New worksheet** and click **Finish**. Excel display the PivotTable designer.
6. Drag the **status** field, followed by the **subsystem** field, from the pallet to the report heading fields as shown in the following figure.



Excel displays a table showing your data summarized by subsystem.

7. Right-click the summarized list and choose **PivotChart** from the context menu. Excel displays a bar chart of your data.
8. To change the chart type, right-click the chart and choose **Chart Type...** from the context menu. To display a pie chart, choose **Pie** from the **Standard Types** tab and specify your preferred display options.

A simple Excel bar chart: bug totals for the last four quarters

This chart displays the number of bugs opened in the each of the last four quarters. To create the chart, perform the following steps:

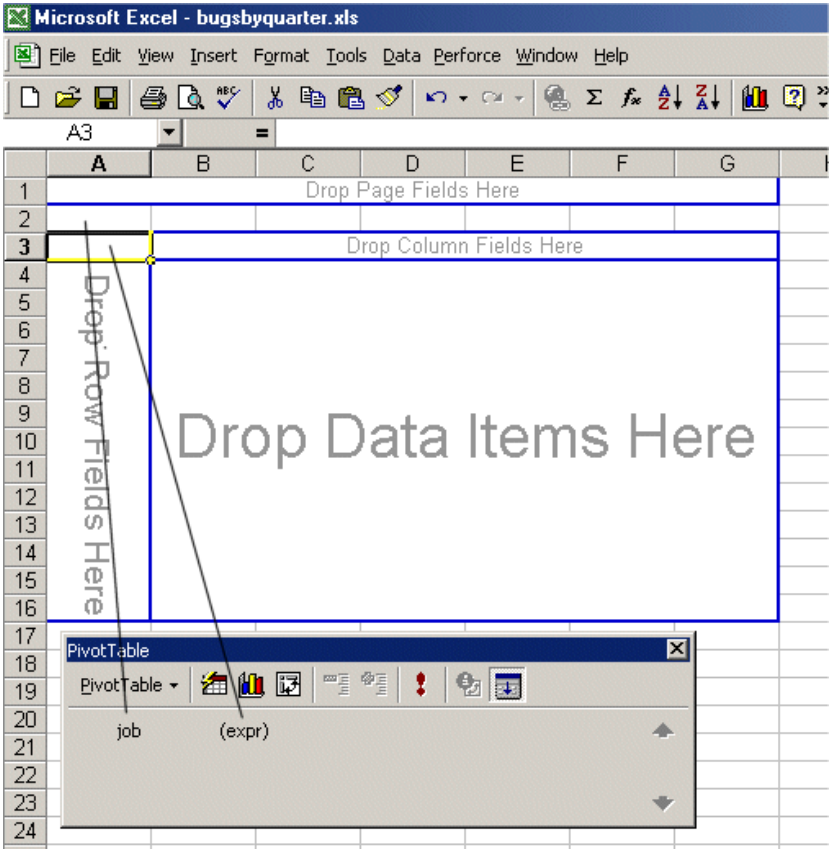
1. Launch Excel. A new, empty spreadsheet is displayed.
2. Choose **Data>Get External Data>New Database Query...** The **Choose Data Source** dialog is displayed.
3. On the **Databases** tab, choose the Perforce data source from the list of data sources and click **OK**. The **Query Wizard - Choose Columns** dialog is displayed.
4. Expand any table, choose any one column, and click **>** to move the column into the Columns in your query list. It does not matter which column you choose, because you will replace the entire query with your own SQL.
5. Click **Next**. The **Filter Data** dialog is displayed.
6. Click **Next**. The **Sort Data** dialog is displayed.
7. Click **Next**. The **Finish Dialog** is displayed.
8. Choose the **View or edit data in Microsoft Query** option and click **Finish**. Microsoft Query is launched, displaying your initial query.
9. Choose **View>SQL...** The **SQL** dialog is displayed.
10. Enter the following SQL query in the SQL window and click **OK**:

```
select jobs.job,
     concat(concat(year(reporteddate), ' '),
     concat('q',quarter(reporteddate))) from jobs jobs
     where (jobs.type='bug')
     and timestampdiff(7,reporteddate,curtimestamp())<3)
     order by 2
```

11. Choose **File>Return Data to Microsoft Excel**. The data is returned to your spreadsheet.

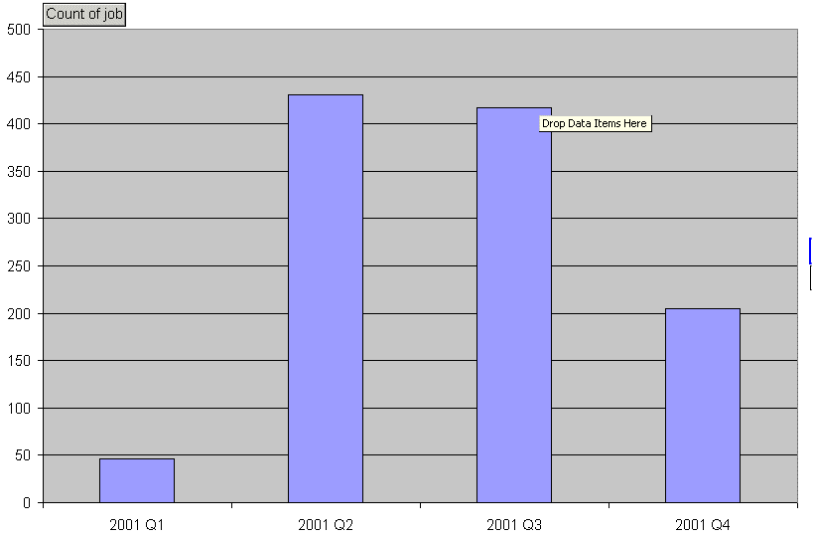
To format the bar chart, perform the following steps:

1. Select the data: click the top right cell where the data starts, scroll to the bottom left cell and shift-click it. Excel highlights the cells you selected.
2. Choose **Data>PivotTable and PivotChart Report**. The PivotTable and PivotChart wizard is launched and displays the **Step 1 of 3** dialog.
3. Select the **Microsoft Excel list or database** and **PivotTable** options and click Next. The **Step 2 of 3** dialog is displayed, listing the range of data in the selected columns.
4. Click Next. The Step 3 of 3 dialog is displayed, asking where you want to put the chart.
5. Choose **New worksheet** and click Finish. Excel displays the PivotTable designer.
6. Drag the **job** field, followed by the **expr** field, from the pallet to the report heading fields as shown in the following figure.



Excel displays a table showing your data summarized by subsystem.

7. Right-click the summarized list and choose **PivotChart** from the context menu. Excel displays a bar chart of your data, such as the example in the following figure.



Chapter 4 Codeline Management

For details about supported functions, refer to Appendix B, P4Report Functions.

Which changelists affect a release?

To list changes without including integrations:

```
select change from changes
     where filespec = '//depot/main/release2.0/...';
```

To list changes including integrations, include the `options='integrated'` clause:

```
select change from changes
     where filespec = '//depot/main/release2.0/...'
     and p4options = 'integrated';
```

How many changes are in this release?

If the release is a branch:

```
select count(*) from changes
     where filespec = '//depot/release1.1/...';
```

If the release is stored in a label:

```
select count(*) from changes
     where filespec = '//...@release1.1';
```

Which changes are in this release?

If the release is a branch:

```
select * from changes
     where filespec = '//depot/release1.1/...';
```

If the release is stored in a label:

```
select * from changes
     where filespec = '//...@release1.1';
```

Which files changed after being branched?

When you integrate files to a new branch, the files in the new branch have a revision number of 1. This query detects files with higher revision numbers by seeking files that have more than more record in the `FILES` table. Using this logic, the following query detects all changed files in `//depot/branch1/...`:

```
select file, count(*) from files
     where file like '//depot/branch1/%' group
     by file having count(*) > 1;
```

Is a particular change in a specified branch?

The answer returned by this query is independent of whether the change was made in a specified branch or integrated from another branch.

```
select fromfile, endfromrev from files, integs
     where change = 123 and
     file = tofile and
     how in ('copy into', 'merge into', 'branch into', 'delete into') and
     fromfile like '//depot/branch1/%' and
     revision between starttorev and endtorev

union

select file, revision from files
     where change = 123 and
     file like '//depot/branch1/%';
```

Appendix A Database Schema and SQL Keywords

P4Report includes an ODBC driver that renders a subset of the information in the Perforce database as a set of tables. This chapter describes the SQL data types of the data returned by P4Report. You can use the information in this chapter to create SQL queries. Note that the P4Report engine adds some virtual columns for query purposes; these columns are not physically present in the Perforce metadata.

This chapter also includes a list of SQL keywords reserved by P4Report.

P4Report System Catalogs

The system catalogs are tables that contain data describing the Perforce metadata tables.

SYSTABS

Lists all tables

Column Name	Data Type	Description
CATALOG	CHAR(32)	The name of the catalog in which the table resides (`system` or `perforce`).
TABNAME	CHAR(32)	The name of the table.

SYSTYPES

Lists all mapped types. This system catalog is included for completeness. The information stored in the `SYSTYPES` table is not of general utility for creating Perforce-related reports.

Column Name	Data Type	Description
TYPE_NAME	CHAR(32)	Name of the data type.
DATA_TYPE	SMALLINT	Corresponding ODBC or driver-specific SQL data type.

Column Name	Data Type	Description
COLUMN_SIZE	INTEGER	• Numeric data types: the maximum precision. • String data types: the length in characters. • Datetime data types: the length of the string representation. • Interval data types: the number of characters in the character representation of the interval. NULL for data types for which column size is not applicable.
LITERAL_PREFIX	CHAR(5)	The characters that precede a literal: • Character data: a single quotation mark (') • Binary data types: 0x NULL for data types that have no prefix.
LITERAL_SUFFIX	CHAR(5)	The characters that follow a literal: • Character data: a single quotation mark (') NULL for data types that have no suffix.
CREATE_PARAMS	CHAR(32)	Lists the parameters that can be specified for each data type.
NULLABLE	SMALLINT	Indicates whether the data type accepts a NULL value.
CASE_SENSITIVE	SMALLINT	Indicates whether a character data type is case-sensitive.
SEARCHABLE	SMALLINT	Specifies how the data type is used in a WHERE clause.
UNSIGNED_ATTRIBUTE	SMALLINT	Indicates whether the data type is unsigned.
FIXED_PREC_SCALE	SMALLINT	Indicates whether the data type has predefined fixed precision and scale.
AUTO_UNIQUE_VALUE	SMALLINT	Indicates whether the data type is autoincrementing.
LOCAL_TYPE_NAME	CHAR(32)	Specifies the localized data source-dependent name of the data type.

Column Name	Data Type	Description
MINIMUM_SCALE	SMALLINT	The minimum scale of the data type. For example, an SQL_TYPE_TIMESTAMP column might have a fixed scale for fractional seconds.
MAXIMUM_SCALE	SMALLINT	The maximum scale of the data type.
SQL_DATA_TYPE	SMALLINT	The SQL data type as it appears in the SQL_DESC_TYPE field of the descriptor.
SQL_DATETIME_SUB	SMALLINT	For SQL_DATETIME or SQL_INTERVAL, the datetime/interval subcode.
NUM_PREC_RADIX	INTEGER	• Approximate numeric types: contains 2 to indicate that COLUMN_SIZE specifies a number of bits. • Exact numeric types: contains the value 10 to indicate that COLUMN_SIZE specifies a number of decimal digits. Otherwise, this column is NULL.
INTERVAL_PRECISION	SMALLINT	For interval data types, contains the value of the interval leading precision.

SYSCOLS

Lists all columns of all tables

Column Name	Data Type	Description
CATALOG	CHAR(32)	The name of the catalog in which the table resides (system or perforce).
TABNAME	CHAR(32)	The name of the table.
COLNAME	CHAR(32)	The name of the column within the table.
COLNO	SMALLINT	The number of the column within the table, starting with 1.
COLTYPE	SQLTYPE	The data type of the column.

SYSIDXS

Lists the primary key for each table.

Column Name	Data Type	Description
CATALOG	CHAR(32)	The name of the catalog in which the table resides (system or perforce).
TABNAME	CHAR(32)	The name of the table.
KEYNAME	CHAR(32)	The name of the key.
COLNAME	CHAR(32)	The name of the column that forms this part of the key.
COLNO	SMALLINT	The position of this column within the key, starting with 1.

P4Report Tables

BRANCHES

Contains a row for each branch specification.

Column Name	Data Type	Description
BRANCH	VARCHAR(254)	Name of branch
OWNER	VARCHAR(254)	The Perforce user that created this branch.
P4OPTIONS	VARCHAR(254)	Branch update options: locked or unlocked.
UPDATED	TIMESTAMP	The date this specification was last modified
ACCESSED	TIMESTAMP	The date of the last integration performed using this branch.
DESCRIPTION	LONGVARCHAR	A short description of the branch (optional).

CHANGES

Contains a row for each changelist. This table cannot be updated using P4Report.

Column Name	Data Type	Description
CHANGE	INTEGER	Changelist number.
DATE	TIMESTAMP	Date the changelist was created.
USER	VARCHAR(254)	Perforce user that created the changelist.
CLIENT	VARCHAR(254)	Perforce client that created the changelist.
STATUS	VARCHAR(15)	pending or submitted

Column Name	Data Type	Description
DESCRIPTION	LONGVARCHAR	User-specified description of the changelist.
P4OPTIONS	VARCHAR(254)	Specifies query options. If you specify multiple options, separate them with spaces, commas, semicolons or colons. Valid values are: • integrated: include any changes integrated into the specified files. Valid only in conjunction with the FILESPEC column. Must be specified in lower case. For example: SELECT * FROM CHANGES WHERE FILESPEC = '//depot/main/release2.0/...' AND P4OPTIONS = 'integrated'; • longdesc: By default, the 31-character short description is returned, to ensure best query performance. To return the full description, specify the longdesc option. Note that queries run more slowly with the longdesc option.
FILESPEC	VARCHAR(254)	The files in the changelist. You can query this column using Perforce wildcards. For example: SELECT * FROM CHANGES WHERE FILESPEC = '//depot/main/doc/...'

CLIENTS

Contains a row for each client

Column Name	Data Type	Description
CLIENT	VARCHAR(254)	Name of the client.
OWNER	VARCHAR(254)	Perforce user that created the client.
P4OPTIONS	VARCHAR(254)	• [no]allwrite • [no]clobber • [no]compress • [un]locked • [no]modtime • [no]rmdir
ROOT	VARCHAR(254)	Client root directory.

Column Name	Data Type	Description
HOST	VARCHAR(254)	The host machine where the client workspace resides.
MAPSTATE	INTEGER	Set to 1 if the client's "have" list (the server's list of file revisions in the client workspace) is in agreement with the client view, or 0 if the "have" list and client view are not in agreement (because the client view has been changed).
UPDATED	TIMESTAMP	Last time the client specification was changed.
ACCESSED	TIMESTAMP	Last time the client performed a Perforce action.
DESCRIPTION	LONGVARCHAR	User-specified description of the client workspace.

COUNTERS

Contains a row for each counter

Column Name	Data Type	Description
COUNTER	VARCHAR(254)	Name of the counter.
VALUE	INTEGER	Value of the counter.

FILES

Contains a row for each revision of each file in the depot. This table cannot be updated using P4Report.

Column Name	Data Type	Description
FILE	VARCHAR(254)	Full file path.
REVISION	SMALLINT	File revision number.
TYPE	VARCHAR(20)	Perforce file type.
ACTION	VARCHAR(10)	Last Perforce action performed on the file.
TIME	TIMESTAMP	Time of last Perforce action.,

Column Name	Data Type	Description
CHANGE	INTEGER	Changelist associated with this revision.
REVSPEC	VARCHAR(254)	Revision specifier column, added by P4Report to support queries using Perforce revision specifications such as change numbers, labels and dates. For example: `select * from files where revspec='2002/12/24'`

FIXES

Contains a row for each fix.

Column Name	Data Type	Description
JOB	VARCHAR(254)	Job number/description (for example `job001234`).
CHANGE	INTEGER	Changelist that fixed this job.
P4OPTIONS	VARCHAR(254)	Specifies query options. Valid values are: • `integrated`: list fixes associated with changelists that affect files which have been integrated into the files specified in the `FILESPEC` column. Valid only in conjunction with the `FILESPEC` column. Must be specified in lower case. For example, to list all fixes affecting files under `branch1`, including those that have been integrated from other files: `SELECT * from fixes WHERE filespec = '//depot/branch1/...' AND options = 'integrated';`
FILESPEC	VARCHAR(254)	Special-purpose column for querying using wildcards; for example: `[...] WHERE FILESPEC = //depot/main/release1.1/...`

GROUPS

Contains a row for each group. Can be filtered by the Perforce users in the group.

Column Name	Data Type	Description
GROUPNAME	VARCHAR(254)	Name of Perforce group.
USERNAME	VARCHAR(254)	User names of group members
SUBGROUP	SMALLINT	Flag: 0 for groups which are not subgroups, 1 for groups which are subgroups.
MAXRESULTS	INTEGER	Maximum number of rows in metadata that queries from group members are allowed to affect.
MAXSCANROWS	INTEGER	Maximum number of rows in metadata that user queries are allowed to scan.
TIMEOUT	INTEGER	Length of time (in seconds) a login ticket remains valid.
USERSPEC	VARCHAR(254)	The Perforce user to be used to qualify groups. For example, to return the groups that contain the Perforce user `joe`, issue the following query: `select * from groups where userspec = 'joe';` Note: The column is returned blank for the query `select * from groups`. You must specify a `where` clause containing the `USERSPEC` column.

INFO

Contains Perforce connection information, one row for each item.

Column Name	Data Type	Description
NAME	VARCHAR(32)	Name of the connection setting (for example, `P4PORT`, `P4USER`, etc.)
VALUE	VARCHAR(254)	Value of the setting.

INTEGS

Contains a row for each file integration. This table cannot be updated using P4Report.

Column Name	Data Type	Description
TOFILE	VARCHAR(254)	Target file.
FROMFILE	VARCHAR(254)	Source file.
STARTTOREV	SMALLINT	Starting revision number of target file.
ENDTOREV	SMALLINT	Ending revision of target file. If integrating from a single revision (as opposed to a revision range), the STARTTOREV and ENDTOREV fields will be identical
STARTFROMREV	SMALLINT	Starting revision number of source file.
ENDFROMREV	SMALLINT	Ending revision of source file. If integrating from a single revision (as opposed to a revision range), the STARTFROMREV and ENDFROMREV fields will be identical
HOW	VARCHAR(15)	Integration method: • merge into • branch into • copy into • ignore into • delete into

JOBS

Contains all jobs. This table is dynamically generated from your Perforce job specification, which you can customize. The following columns are always appended to this table:

Column Name	Data Type	Description
P4OPTIONS	VARCHAR(254)	Specifies query options. Valid values are: • integrated: list jobs associated with changelists that affect files which have been integrated into the files specified in the FILESPEC column. Valid only in conjunction with the FILESPEC column. Must be specified in lower case. For example, to list jobs that have fixes affecting files under branch1, including files that have been integrated from other files. `SELECT * from jobs WHERE filespec = '//depot/branch1/...' AND options = 'integrated';`
FILESPEC	VARCHAR(254)	Special-purpose column for querying using wildcards; for example: `WHERE FILESPEC = //depot/main/release1.1/...`

LABELS

Lists all labels

Column Name	Data Type	Description
LABEL	VARCHAR(254)	Name of the label
OWNER	VARCHAR(254)	Perforce user that created the label.
P4OPTIONS	VARCHAR(254)	Locked or unlocked
UPDATED	TIMESTAMP	Last time the label definition was changed.
ACCESSED	TIMESTAMP	Last time the label was used to perform a Perforce action.
DESCRIPTION	LONGVARCHAR	User-specified description of the label.

OPENED

Contains a record for each opened file

Column Name	Data Type	Description
DEPOTFILE	VARCHAR(254)	Depot path of the open file.
CLIENTFILE	VARCHAR(254)	Client location of the open file.
REVISION	SMALLINT	Revision number of the open file
ACTION	VARCHAR(10)	Indicates open for `add`, `edit`, `delete`, `branch`, `integrate`, or `import`.
CHANGE	INTEGER	Number of changelist in which the file is open for the specified action.
TYPE	VARCHAR(20)	Perforce file type (if being changed).
USER	VARCHAR(254)	Perforce user that opened the file.
CLIENT	VARCHAR(254)	Perforce client that opened the file.

PROTECTIONS

Lists Perforce protections. Only the Perforce superuser can access this table.

Column Name	Data Type	Description
MODE	VARCHAR(10)	The Perforce permission: • `list` • `read` • `review` • `open` • `write` • `super`
TYPE	VARCHAR(10)	The type of recipient of the permission: `group` or `user`.
NAME	VARCHAR(254)	A Perforce group or username.
HOST	VARCHAR(254)	The IP address of the client host.
PATH	VARCHAR(254)	The part of the depot to which the permission applies.

USERS

Lists all Perforce users

Column Name	Data Type	Description
USER	VARCHAR(254)	Name of Perforce user.
EMAIL	VARCHAR(254)	Email of user.
FULLNAME	VARCHAR(254)	User-specified name of user.
UPDATED	TIMESTAMP	Last time user definition was changed.
ACCESSED	TIMESTAMP	Last time this user performed a Perforce action.

SQL Keywords

ALL	INTO
AND	IS
ANY	JOIN
AS	LEFT
ASC	LIKE
AVG	LOCAL
BETWEEN	MATCH
BY	MAX
CAST	MIN
CORRESPONDING	NATURAL
COUNT	NOT
CREATE	NULL
CROSS	ON
DELETE	OR
DESC	ORDER
DISTINCT	OUTER
DROP	RIGHT
ESCAPE	SELECT
EXCEPT	SET
EXISTS	SOME
FALSE	SUM
FROM	TABLE
FULL	TEMPORARY
GLOBAL	TRUE
GROUP	UNION
HAVING	UNIQUE
IN	UNKNOWN
INNER	UPDATE
INSERT	USING
INTERSECT	VALUES
	WHERE

Appendix B P4Report Functions

Connection Information Functions

The following table lists functions that return Perforce connection information.

Function	Description
`P4CLIENT()`	Returns the client specification currently in effect.
`P4HOST()`	Returns the name of the client host computer currently in effect, if specified in the definition of the client workspace.
`P4PORT()`	Returns the server/port setting currently in effect.
`P4USER()`	Returns the name of the Perforce user currently in effect.

Use the functions as shown in the following example:

```
select p4host(), p4client(), p4port(), p4user() from users where
user='jones';
```

String Functions

The following table lists the standard SQL string functions supported by P4Report.

Function	Description
`ASCII(`*`string`*`)`	Returns the ASCII code value of the leftmost character of *`string`* as an integer.
`CHAR(`*`code`*`)`	Returns the character that has the specified ASCII value.
`CHAR_LENGTH(`*`string`*`)` `CHARACTER_LENGTH(`*`string`*`)`	Returns the number of characters of the string.
`CONCAT(`*`string1`*`, `*`string2`*`)`	Returns a character string that is the result of concatenating *`string2`* to *`string1`*.
`INSERT(`*`string1`*`, `*`start`*`, `*`length`*`, `*`string2`*`)`	Returns a character string where *`length`* characters have been deleted from *`string1`*, beginning at *`start`*, and where *`string2`* has been inserted into *`string1`*, beginning at *`start`*.

Function	Description
`LCASE(string)`	Returns the specified string with all uppercase characters converted to lowercase.
`LEFT(string, count)`	Returns the leftmost *count* characters of *string*.
`LOCATE(string1, string2[, start])`	Returns the starting position of the first occurrence of *string1* within *string2*. The search for the first occurrence of *string1* begins with the first character position in *string2* unless *start* is specified. If *start* is specified, the search begins with the character position indicated by the value of *start*. The first character position in *string2* is 1. If *string1* is not found within *string2*, 0 is returned.
`LTRIM(string)`	Returns the characters of *string*, with leading white space removed.
`REPEAT(string, count)`	Returns a character string composed of *string* repeated *count* times.
`REPLACE(string1, string2, string3)`	Search *string1* for occurrences of *string2* and replace with *string3*.
`RIGHT(string, count)`	Returns the rightmost *count* characters of *string*.
`RTRIM(string)`	Returns the characters of *string* with trailing blanks removed.
`SPACE(count)`	Returns a character string consisting of *count* spaces.
`SUBSTRING(string, start, length)`	Returns a character string that is derived from *string*, beginning at the character position specified by *start* for *length* characters.
`UCASE(string)`	Returns a string equal to that in *string*, with all lowercase characters converted to uppercase.

Numeric Functions

The following table lists the standard SQL numeric functions supported by P4Report.

Function	Description
`ABS(numeric)`	Returns the absolute value of *numeric*.
`ACOS(numeric)`	Returns the arccosine of *numeric* as an angle, expressed in radians.
`ASIN(numeric)`	Returns the arcsine of *numeric* as an angle, expressed in radians.
`ATAN(numeric)`	Returns the arctangent of *numeric* as an angle, expressed in radians.
`ATAN2(numeric1, numeric2)`	Returns the arctangent of the x and y coordinates, specified by *numeric1* and *numeric2*, respectively, as an angle expressed in radians.
`CEILING(numeric)`	Returns the smallest integer greater than or equal to *numeric*.
`COS(numeric)`	Returns the cosine of *numeric*.
`DEGREES(numeric)`	Returns the number of degrees converted from *numeric* radians.
`EXP(numeric)`	Returns the exponential value of *numeric*.
`FLOOR(numeric)`	Returns the largest integer less than or equal to *numeric*.
`LOG(numeric)`	Returns the natural logarithm of *numeric*.
`LOG10(numeric)`	Returns the base 10 logarithm of *numeric*.
`MOD(numeric1, numeric2)`	Returns the remainder (modulus) of *numeric1* divided by *numeric2*.
`PI()`	Returns the constant value of pi.
`RADIANS(numeric)`	Returns the number of radians converted from *numeric* degrees.
`RAND([numeric])`	Returns a random floating-point value, optionally using *numeric* as the seed value.
`SIGN(numeric)`	Returns an indicator of the sign of *numeric*. If *numeric* is less than zero, –1 is returned. If *numeric* equals zero, 0 is returned. If *numeric* is greater than zero, 1 is returned.
`SIN(numeric)`	Returns the sine of *numeric*.

Function	Description
SQRT(*numeric*)	Returns the square root of *numeric*.
TAN(*numeric*)	Returns the tangent of *numeric*, where *numeric* is an angle expressed in radians.

Date and Time Functions

The following table lists the standard SQL date and time functions supported by P4Report.

Function	Description
CURDATE() CURRENT_DATE()	Returns the current date.
CURTIME() CURRENT_TIME()	Returns the current local time.
CURTIMESTAMP() CURRENT_TIMESTAMP() NOW()	Returns the current local date and local time.
DAYNAME(*date*)	Returns a character string containing the name of the day.
DAYOFMONTH(*date*)	Returns the day of the month based on the month field in *date* as an integer value in the range of 1–31.
DAYOFWEEK(*date*)	Returns the day of the week based on the week field in *date* as an integer value in the range of 1–7, where 1 represents Sunday.
DAYOFYEAR(*date*)	Returns the day of the year based on the year field in *date* as an integer value in the range of 1–366.
HOUR(*time*)	Returns the hour based on the hour field in *time* as an integer value in the range of 0–23.
MINUTE(*time*)	Returns the minute based on the minute field in *time* as an integer value in the range of 0–59.
MONTH(*date*)	Returns the month based on the month field in *date* as an integer value in the range of 1–12.
MONTHNAME(*date*)	Returns a character string containing the name of the month.
QUARTER(*date*)	Returns the quarter in *date* as an integer value in the range of 1–4, where 1 represents January 1 through March 31.

Function	Description
`SECOND(time)`	Returns the second based on the second field in *`time`* as an integer value in the range of 0–59.

`TIMESTAMPADD(unit, qty, sourcetime)`	Modifies a timestamp by adding or subtracting time.
	The *`unit`* parameter specifies the unit of time to be added as follows:
	1 = seconds
	2 = minutes
	3 = hours
	4 = days
	5 = weeks
	6 = months
	7 = quarters
	8 = years
	The *`qty`* parameter specifies how many units (minutes, days or months etc.) are to be added. To subtract time, specify a negative value.
	The *`sourcetime`* parameter specifies the timestamp expression to be modified.
	Examples:
	`TIMESTAMPADD(1, 10, '1997-10-10 12:00:00')`
	Result: `1997-10-10 12:00:10`
	`TIMESTAMPADD(3, 10, '1997-10-10 12:00:00')`
	Result: `1007-10-10 22:00:00`

`TIMESTAMPDIFF(interval, tstamp1, tstamp2)`	Returns the integer number of intervals of type *`interval`* by which *`tstamp2`* is greater than *`tstamp1`*. Specify *`interval`* as an integer indicating the desired unit, as follows: 1 = seconds 2 = minutes 3 = hours 4 = days 5 = weeks 6 = months 7 = quarters 8 = years
`YEAR(date)`	Returns the year based on the year field in *`date`* as an integer value.

Appendix C Sample Crystal Report Queries

The following table lists the SQL used in the sample compiled Crystal Reports included with P4Report. Note that some of the SQL uses ODBC-specific syntax.

Report Description	SQL

ChangesByMonth	`SELECT changes."change", changes."date", changes."description"`
	`WHERE ("changes"."date">={ts '2006-06-01 00:00:00'} AND "changes"."date"<{ts '2006-07-01 00:00:00'})`
	`FROM "changes" changes`

ChangesByUser	`SELECT changes."change", changes."date", changes."user", changes."client", changes."description"`
	`FROM "changes" changes`
	`ORDER BY changes."user" ASC, changes."change" DESC`

ClientsByUser	`SELECT clients."client", clients."owner", clients."root", clients."description"`
	`FROM "clients" clients`
	`ORDER BY clients."owner" ASC`

ClientsNotUsed	`SELECT clients."client", clients."owner", clients."accessed", clients."description"`
	`FROM "clients" clients`
	`WHERE timestampdiff(8, clients."accessed", now()) > 1`
	`ORDER BY clients."owner" ASC`

JobChanges	`SELECT fixes."job", changes."change", changes."date", changes."user", changes."description"`
	`FROM { oj "fixes" fixes`
	`INNER JOIN "changes" changes`
	`ON fixes."change" = changes."change"}`

ModsPastWeek	`SELECT files."file", files."time"`
	`FROM "files" files`
	`WHERE timestampdiff(5, files."time", now()) < 1`
	`ORDER BY 2`

Report Description	SQL

MostChangedFiles	`SELECT files."file"`
	`FROM "files" files`
	`ORDER BY files."file" ASC`

OpenedFilesByUser	`SELECT opened."clientfile", opened."revision", opened."action", opened."change", opened."type", opened."user"`
	`FROM "opened" opened`
	`ORDER BY opened."user" ASC`

Index