# The Open Personal Data Standard

## SQL Schema Reference Guide

## Table Of Contents

# 1 Introduction

## 1.1 This Document

This document describes elements of the Open Personal Data Standard (ODPS). The ODPS is a proposed standard to allow different Personal Information Managers(PIM) to access and share data in a uniform manner.

The scope is to provide a single SQL database that allows any application to store data in a flexible and retrievable  fashion. There are no restrictions on the type of data that can be stored within the structure or the format used. Once a data element has been added to this system any compatible application is able to find and reference this data, even if an application is not able to understand the format this data is stored in. It is still able to reference the data and pass it into other applications.

This is considered a living document and open to review and changes as the subject it describes changes.

## 1.2 Copyright

# 2 SQL Schema Reference Guide

The scope is to provide a single SQL database that allows any application to store data in a flexible and retrievable fashion. There are no restrictions on the type of data that can be stored within the structure or the format used. Once a data element has been added to this system any compatible application is able to find and reference this data, even if an application is not able to understand the format this data is stored in. It is still able to reference the data and pass it into other applications.

## 2.1 Features

The following list of features are built into the OPDS schema design.

- Easy to implement.
- Extendability.
- Multiple Host Synchronization Support.
- Linking between different data elements within the database.
- Multi level data grouping.
- Secondary data locations.
- Multiple data formats.
- Default application registration.

# 3 ODPS Tables

## 3.1 ODPS_RESOURCE Table

The ODPS_RESOURCE Table provides a uniform way of identifying data elements across multiple ODPS applications.

```
-- ** ODPS_RESOURCE **
-- Maps a URN to a Standard Resource Tables

CREATE TABLE ODPS_RESOURCE (
  urn    int NOT NULL UNIQUE, -- Unique Resource Number.
  name          text,         -- Display name for resource.
  srt_id        text,         -- Standard Resource Table ID.
  last_modified date,         -- The date and time of the last modification

  PRIMARY KEY(group_id)
);
```

## 3.2 ODPS_LINK

The ODPS_LINK Table enables applications to register a link between multiple URNs. The meaning of this link is dependent of the context.

For example:

- A calender application would be able to link an event with a collection of contacts.
- A Banking application could link a transaction to a contact and/or a calender event.
- A Contact manager could provide the option to link contacts by there relationships

The ODPS_LINK stores the source and destination URNs within a single table row.

```
-- ** ODPS_LINK **
-- Maps links between multiple URNs.

CREATE TABLE ODPS_LINK (
  source_urn       text NOT NULL, -- Resource Linking.
  destination_urn  text NOT NULL, -- Linked Resource.
  description      text,          -- Description of link.
  PRIMARY KEY( source_urn, destination_urn )
);
```

## 3.3 ODPS_GROUP Table

Groups are used to provide a great deal of flexibility within the ODPS dataset. A single resource may be a member of any number of different groups.

The ODPS schema uses groups to identify sync hosts, it is permissible for a resource to be associated with multiple Sync Groups allowing it to be synced on multiple hosts. But care must be taken to ensure the host concerned is able to cope with the data type being synced.

For example there is little point trying to send a calender entry to a phone book application on a mobile phone. The ODPS_SYNC_HOSTS table allows for a list of valid resource types with each host.

Groups can also be used by applications to separate resources into different categories for example: Business and Personal etc. The same group names can be used for multiple resource types such as contacts appointments and banking transactions.

Each of the above options can be combined as your application chooses or simply ignored. Apart from the ability to specify sync hosts there is no requirement for your application to use groups at all.

```
-- ** ODPS_GROUP **
-- Provides a Name and description link
-- for each group within the ODPS dataset

CREATE TABLE ODPS_GROUP (
  group_id     int NOT NULL UNIQUE,  -- Group Identifier
  parent_group int,                  -- Parent Group id
  name         text NOT NULL UNIQUE, -- Display name
  description  text,                 -- Brief Description

  PRIMARY KEY(group_id)
);
```

## 3.4 ODPS_GROUP_MEMBER Table

The ODPS_GROUP_MEMBER Table links a ODPS Unique Resource Number (URN) with a Group Identifier. A single URN may be a member of multiple groups. Each membership would be identified as a different row within the ODPS_GROUP_MEMBER table

```
-- ** ODPS_GROUP_MEMBER **
-- Links a URN with a group_id.
-- A single URN may be a member of multiple
-- groups each membership should be on a
-- separate row

CREATE TABLE  ODPS_GROUP_MEMBER (
  urn       int NOT NULL, -- The ODPS Unique Resource Number
  group_id  int NOT NULL, -- Group Identifier
);
```

## 3.5 ODPS Standard Resource Tables

A ODPS Standard Resource Table is used to store actual data elements within the ODPS Database Schema.

These tables have an extremely simple structure. Each row within the table represents a single field within the resource. For example Name, Date of birth, Telephone Number or Zip Code for a contact manager.

Each row is assigned to a valid Unique Resource Number allowing elements to be grouped together.

A element_name field is supplied to allow easy searching of the database.

The entire data for each element is then stored within the element_value field.

```
-- ** TABLE_NAME **
-- Table Names should be in Upper Case while Short and descriptive.
-- Examples are CONTACT EVENT TRANSACTION
CREATE TABLE TABLE_NAME (
  urn           int NOT NULL,  -- The ODPS Unique Resource Number
  element_name  text NOT NULL, -- Name used for identity and display
  element_value text NOT NULL  -- formated element data.

  PRIMARY KEY(urn)
);
```

Most data formats can be easily stored within this structure.

The following tables provide examples of how to use this structure to store a single contacts resource first using the Internet Mail Consortium's vCard format followed by XML. These formats are provided only as examples of how to include data within the OPDS table structure. The formats used are described within different documents.

| URN | element_name | element_value |
|---|---|---|
| 101 | BEGIN | BEGIN:VCARD |
| 101 | Name | N:Hall;David;Alan;Mr;BSc |
| 101 | Formatted Name | FN:Mr Dave A Hall BSc |
| 101 | Telephone Number | TEL;HOME:+1-789-4561-2313 |
| 101 | Telephone Number | TEL;PREF;WORK:+1-789-654-3131 |
| 101 | Telephone Number | TEL;CELL:+1-654-568-9854 |
| 101 | Delivery Address | ADR;WORK;POSTAL:;Suite 101;234 A Street;Some Town;GA;10101-1111;USA |
| 101 | Delivery Address | ADR;HOME;POSTAL:;;123 Another Street;This Town;CA;20200;USA |
| 101 | END | END:VCARD |

A description of the vCard format can be found at http://www.imc.org/pdi/

| URN | element_name | element_value |
|---|---|---|
| 101 | HEADER | <contact> |
| 101 | First | <first>David</first> |
| 101 | Last | <last>Hall</last> |
| 101 | Other | <other>Alan</other> |
| 101 | Phone | <phone >+1-789-4561-2313</phone> |
| 101 | Address | <address>123 Another Street,This Town,CA,20200,USA</address> |
| 101 | FOOTER | </contact> |

It is worth noting that when dealing with SQL databases there is no concept of data order, other than that imposed by the programmer using the ORDER BY clause. With this in mind it will be the responsibility of the developer to enforce any ordering their application may require when retrieving this data.

It would also be sensible not to store header or footer tags within the database. As this could be easily inferred by the application at run time, depending on the format used.

The ODPS Database structure provides the following list of Standard Resource Tables by default.

## 3.6 ODPS_RESOURCE_TABLE

This table is used to provide a list of all Standard Resource Tables available within the current Environment.

The srt_id field is a unique table identifier. The use of  srt_ids below 1000 are reserved for ODPS standard tables.

The table_name field should match the exact name used within the database. Each table name only needs to be unique within the location specified.

The type field is used to identify the purpose of the table. Valid values include Contact, Event or Note. Developers have to option to create tables of there own type matching the SRT format.

The description is used to provide a breif explanation of the tables purpose.

The format filed is used to identify the data format used within this table. This field contains a format_id from the ODPS_FORMATS table.

The location field provides the ability to locate a SRT within a different file.

```
-- ** ODPS_RESOURCE_TABLE **
-- This table is used to identify a list
-- Of available Standard Resource Tables
CREATE TABLE ODPS_RESOURCE_TABLE (
  srt_id        int NOT NULL UNIQUE, -- The Standard Resource Table ID
  table_name    text NOT NULL,       -- Table name must be unique within location
  type          text NOT NULL,       -- The type of this table IE
                                     -- ^ [Contacts|Events|Notes]
  location      text,                -- URL of database used to store this table.
  format_id     text NOT NULL,       -- Data Format used for this table.
  description   text,                -- Brief description of purpose.

  PRIMARY KEY(srt_id)
);
```

## Standard Entries

The following standard entries are provided within the ODPS Database Schema. Empty Standard Resource Tables are provided for each of these Enteries

| srt_id | table_name | type | description | format_id | location |
|---|---|---|---|---|---|
| 1 | CONTACT | Contacts | Address Book entries | VCARD | NULL |
| 2 | EVENT | Events | Calender entries | VCAL | NULL |
| 3 | NOTE | Notes | Notes | HTML | NULL |
| 4 | TODO | To-dos | To Do list entries | VCAL | NULL |

## 3.7 ODPS_FORMAT

The ODPS_FORMAT table is used to list all formats used within Standard Resource Tables. This table simply links a format id to name and short description of the format. Currently the only value to this table is to allow your application to display short descriptions of the available formats etc.

The application field indicates an executable program to use as a default for displaying this format type. This application must be able to receive a URN via the command line.

This table will be extended as the Data Conversion Tools are created. These tools will have the option of looking for format information in this table.

```
-- ** ODPS_FORMAT **
-- This table is used to identify a list
-- Of used data formats
CREATE TABLE ODPS_FORMAT (
  format_id     text NOT NULL UNIQUE, -- Upper case identity string
  format_name   text NOT NULL,        -- Display Name of format.
  description   text,                 -- Brief description of purpose.
                                      -- ^ a NULL entry indicate this db file.
  application   text,                 -- Default display application.

  PRIMARY KEY (format_id)
);
```

## Standard Entries

The following standard entries are provided within the ODPS Database Schema.

| format_id | format_name | description |
|---|---|---|
| OPDSC | Contacts | Open Personal Data Standard Contacts Format |
| OPDSE | Events | Open Personal Data Standard Events Format |
| OPDSB | Banking | Open Personal Data Standard Banking Format |
| VCARD | vCard | The vCard Electronic Business Card standard. Please see http://www.imc.org/pdi/ for further detail. |
| VCAL | vCalendar | The vCalendar Cross-Platform Scheduling standard. Please see http://www.imc.org/pdi/ for further detail. |
| NVP | Name Value Pair | A simple Name=Value pair protocol. |
| TXT | Plain Text | Simple text string. UTF-8 ? |
| HTML | HTML | Subset of HTML to allow flexible formatting. |

## 3.8 ODPS_SYNC_HOST

The data storage requirements for synchronization are fairly simple to implement.

A dedicated Sync table should be stored along with a table linking group ids to server hosts.

The Sync Application described within the ODPS structure would then be able to select a list of data elements based on the groups linked to this host.

To allow each element to be a member of multiple groups and each host to have the option of Syncing with multiple groups  The algorithm for selecting data elements should be something Functionally equivalent to

```
SELECT
  URN
FROM
  ODPS_RESOURCE
WHERE group_id in (SELECT
                     group_id
                   FROM
                     ODPS_SYNC_GROUP
                   WHERE host_id=$host_id);
```

For the sake of speed  I would advise circumventing the inner select within this statement and instead parsing a simple string of previously selected group_ids.

As the ODPS_RESOURCE table store the details for resources of multiple types it is also necessary to ensure that that the Sync Application is only pulling resources that are able to be passed to the host concerned. For Example:

```
SELECT
  URN
FROM
  ODPS_RESOURCE
      WHERE type in ($host_types)
      AND group_id in ($group_list);
```

Again using a previously selected list of types from the ODPS_SYNC_HOST table and parsing it into a string.

```
-- ** ODPS_SYNC_HOST Table **
-- This table is used to store information
-- required to connect to remote hosts
-- This tables definition is far from
-- finished further data will be required
-- for an effective Syncing Application

CREATE TABLE ODPS_SYNC_HOST (
  host_id         int NOT NULL,  -- Unique identifier for host system.
  host_name       text NOT NULL, -- Unique name for host system.
  host_auth       text,          -- Data required for host authorization.
                                 -- ^ Encryption should be handled by the Sync
                                 -- ^Client
  last_anchor     text,          -- Last Sync Anchor used for sanity checking
  next_anchor     text,          -- Next Sync Anchor used for sanity checking
  conflict_method text,          -- Method used to resolve conflicts
  host_protocols  text,          -- Protocol list understood by host CSV
  host_types      text,          -- List of ODPS Types accepted by this host.
  last_sync       date,          -- The Time and date of the most recent sync.

  PRIMARY KEY(host_id, host_name)
);
```

## 3.9 ODPS_SYNC_GROUP

The ODPS_SYNC_GROUP table is used to link a group to a specific host server.

```
-- ** ODPS_SYNC_GROUP Table **
-- This table is used to store information linking groups
-- of data resources with hosts to be synced. A Sync Host
-- may have multiple groups and a group may be synced with
-- multiple hosts.
-- When the Sync Application is run on a specific host it
-- is required to select all groups from ODPS_SYNC_GROUP
-- that link to the specified host_id then use the
-- ODPS_GROUP_MEMBER table to pass data for each matching
-- element from that group.

CREATE TABLE ODPS_SYNC_GROUP (
  host_id    int NOT NULL, -- Remote host id from ODPS_SYNC_HOST
  group_id   int NOT NULL, -- group id from ODPS_GROUP_MEMBER

  PRIMARY KEY(host_id, group_id)
);
```

# 4 ODPS Database Schema Definition

```
-- ODPS Database Schema v0.2.0
-- This is a proposed schema for storing ODPS Data
-- Much of the design is based on the ODPS PIM Schema v0.1.0
-- composed by Owen Cliffe <occ@cs.bath.ac.uk> 25/3/2002
-- as found on http://ODPS.handhelds.org/pim.sql
-- discussion should take place on the ODPS list <ODPS@handhelds.org>
-- This schema includes structures for storing PIM Configuration and
-- Sync Data to be shared by ODPS Applications
-- Composed by Dave A Hall dave@AdelieSolutions.com


-- ***************************************************
-- *               Resource Tables                   *
-- * The following tables provide the ability to     *
-- * define groups ODPS Resources                     *
-- ***************************************************

-- ** ODPS_RESOURCE **
-- Maps a URN to a Standard Resource Tables

CREATE TABLE ODPS_RESOURCE (
  urn    int NOT NULL UNIQUE, -- Unique Resource Number.
  name   text,                -- Display name for resource.
  srt_id text,                -- Standard Resource Table ID.
  last_modified date,         -- The date and time of the last modification.

  PRIMARY KEY(srt_id)
);

-- ** ODPS_LINK **
-- Maps links between multiple URNs.

CREATE TABLE ODPS_LINK (
  source_urn       text NOT NULL, -- Resource Linking.
  destination_urn  text NOT NULL, -- Linked Resource.
  description      text,          -- Description of link.
  PRIMARY KEY( source_urn, destination_urn )
);

-- ** ODPS_RESOURCE_TABLE **
-- This table is used to identify a list
-- Of available Standard Resource Tables
CREATE TABLE ODPS_RESOURCE_TABLE (
  srt_id        int NOT NULL UNIQUE, -- The Standard Resource Table ID
  table_name    text NOT NULL,       -- Table name must be unique within location
  type          text NOT NULL,       -- The type of this table IE
                                     -- ^ [Contacts|Events|Notes]
  format_id     text NOT NULL,       -- Data Format used for this table.
  description   text                 -- Brief description of purpose.
);


-- ** ODPS_FORMAT **
-- This table is used to identify a list
-- Of used data formats
CREATE TABLE ODPS_FORMAT (
  format_id     text NOT NULL UNIQUE, -- Upper case identity string
  format_name   text NOT NULL,        -- Display Name of format.
  description   text,                 -- Brief description of purpose.
  application   text,                 -- Default display application.
                                      -- ^ a NULL entry indicate this db file.
  PRIMARY KEY (format_id)
);

INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('OPDSC', 'Contacts', 'Open Personal Data Standard Contacts
Format');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('OPDSE', 'Events', 'Open Personal Data Standard Events
Format');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('OPDSB', 'Banking', 'Open Personal Data Standard Banking
Format');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('VCARD', 'vCard', 'The vCard Electronic Business Card
standard.');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('VCAL', 'vCalendar', 'The  vCalendar Electronic Business Card
standard.');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('NVP', 'Name Value Pair', 'A simple Name=Value pair
protocol.');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('TXT', 'Plain Text', 'Simple text string. UTF-8 ?');
INSERT INTO ODPS_FORMAT(format_id, format_name, description)
                VALUES('HTML', 'HTML', 'Subset of HTML to allow flexible
formatting.');

-- ***************************************************
-- *               Group Tables                      *
-- * The following tables provide the ability to     *
-- * define groups to allocate ODPS Resources to them *
-- ***************************************************

-- ** ODPS_GROUP **
-- Provides a Name and description link
-- for each group within the ODPS dataset

CREATE TABLE ODPS_GROUP (
  group_id     int NOT NULL UNIQUE,  -- Group Identifier
  parent_group int,                  -- parent group id.
  name         text NOT NULL UNIQUE, -- Display name
  description  text,                 -- Brief Description

  PRIMARY KEY(group_id)
);

-- ** ODPS_GROUP_MEMBER **
-- Links a URN with a group_id.
-- A single URN may be a member of multiple
-- groups each membership should be on a
-- separate row

CREATE TABLE  ODPS_GROUP_MEMBER (
  urn       int NOT NULL, -- The ODPS Unique Resource Number
  group_id  int NOT NULL  -- Group Identifier
);

-- *********************************************
-- *          Synchronization Tables           *
-- * The following tables are dedicated to the *
-- * Synchronization of ODPS Resources          *
-- *********************************************

             -- ** ODPS_SYNC_HOST Table **
             -- This table is used to store information
             -- required to connect to remote hosts
             -- This tables definition is far from
             -- finished further data will be required
             -- for an effective Syncing Application

             CREATE TABLE ODPS_SYNC_HOST (
          host_id         int NOT NULL, -- Unique identifier for host system.
          host_name       text NOT NULL,-- Unique name for host system.
          host_auth       text,         -- Data required for host authorization.
                                     -- ^ Encryption should be handled by the Sync
                                     -- ^ Client
               last_anchor     text,          -- Last Sync Anchor used for sanity
checking
               next_anchor     text,          -- Next Sync Anchor used for sanity
checking
               conflict_method text,          -- Method used to resolve conflicts
               host_protocols  text,          -- Protocol list understood by host CSV
               host_types      text,          -- List of ODPS Types accepted by this
host.
               last_sync       date,          -- The Time and date of the most recent
sync.

               PRIMARY KEY(host_id, host_name)
             );

             -- ** ODPS_SYNC_GROUP Table **
             -- This table is used to store information linking groups
             -- of data resources with hosts to be synced. A Sync Host
             -- may have multiple groups and a group may be synced with
             -- multiple hosts.
             -- When the Sync Application is run on a specific host it
             -- is required to select all groups from ODPS_SYNC_GROUP
             -- that link to the specified host_id then use the
             -- ODPS_GROUP_MEMBER table to pass data for each matching
             -- element from that group.

             CREATE TABLE ODPS_SYNC_GROUP (
          host_id    int NOT NULL, -- Remote host id from ODPS_SYNC_HOST
          group_id   int NOT NULL, -- group id from ODPS_GROUP_MEMBER

               PRIMARY KEY(host_id, group_id)
             );


-- *********************************************
-- *        Standard Resource Tables           *
-- * The following tables are dedicated to the *
-- * Storage of ODPS Data elements              *
-- *********************************************

-- ** CONTACT **
CREATE TABLE CONTACT (
  urn           int NOT NULL,  -- The ODPS Unique Resource Number
  element_name  text NOT NULL, -- Name used for identity and display
  element_value text NOT NULL, -- formated element data.

  PRIMARY KEY(urn)
);

INSERT INTO ODPS_RESOURCE_TABLE(srt_id, table_name, type, format_id, description)
                        VALUES(1, 'CONTACT', 'Contacts', 'OPDSC',
                               'Address Book entries');

-- ** EVENT **
CREATE TABLE EVENT (
  urn           int NOT NULL,  -- The ODPS Unique Resource Number
  element_name  text NOT NULL, -- Name used for identity and display
  element_value text NOT NULL, -- formated element data.

  PRIMARY KEY(urn)
);

INSERT INTO ODPS_RESOURCE_TABLE(srt_id, table_name, type, format_id, description)
                        VALUES(2, 'EVENT', 'Events', 'OPDSE',
                               'Calendar entries');

-- ** NOTE **
CREATE TABLE NOTE (
  urn           int NOT NULL,  -- The ODPS Unique Resource Number
  element_name  text NOT NULL, -- Name used for identity and display
  element_value text NOT NULL, -- formated element data.

  PRIMARY KEY(urn)
);

INSERT INTO ODPS_RESOURCE_TABLE(srt_id, table_name, type, format_id, description)
                        VALUES(3, 'NOTE', 'Notes', 'HTML',
                               'Notes');

-- ** TODO **
CREATE TABLE TODO (
  urn           int NOT NULL,  -- The ODPS Unique Resource Number
  element_name  text NOT NULL, -- Name used for identity and display
  element_value text NOT NULL, -- formated element data.

  PRIMARY KEY(urn)
);

INSERT INTO ODPS_RESOURCE_TABLE(srt_id, table_name, type, format_id, description)
                        VALUES(4, 'TODO', 'To-Dos', 'OPDSE',
                               'To-Do list entries');
```