IRIS Publication Management System - the first steps towards realization

E.D. Schabell ericks@cs.ru.nl
H.A. Proper erikp@cs.ru.nl
Th.P. van der Weide tvdw@cs.ru.nl

University of Nijmegen, Computing Science Institute, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

1 Introduction

The *IRIS Publication Management System (PMS)* has been a long time in coming. It has been a wish of the IRIS department to have a single entry point for dealing with the publications created by its members. The complexities of not only accepting new submissions, but to process these submissions on through the existing institutional publication infrastructure is not a hurdle easily taken.

The submission of both internal and external publications generates not only a collection of documents, but also a very valuable and maybe useful repository of publication data. If this data were to be collected, protected from inconsistencies and properly organized then one would only be limited in her imagination as the the uses that it could be put. To start with, one can begin to provide a very interesting playground for departmental retrieval experiments, provide various forms of exportable formats (think of HTML, BiBTeX, text, etc.) and generate any type of organizational reporting as deemed necessary (such as yearly overviews of departmental publications).

This document discusses the definition and design of the *IRIS PMS* This includes the motivation (*why*), the (functional) requirements (*what*), the key design principles as well as the actual design (*how*).

1.1 Motivation

In our day-to-day research activities, publications play a key role. Be it publications by ourselves or be it publications by others, having an *IRIS PMS* can potentially bring the following benefits:

Tracking It would allow for improved tracking of our own publications, including archiving of a publication's source files (LaTeX, OpenOffice, ClosedOffice[1], or other formats.)

Reporting Yearly publication records can be generated automatically from the PMS.

Dissemination Our own publications can be made available on the Web automatically from the *IRIS PMS*. This can either be part of an IRIS member's home page, or the home page of the IRIS group.

Referring Centralized storage and maintenance of PDF files and meta-data (e.g. BibTeX data) of publications we refer to in our work. A relevant PDF file needs to be downloaded only once, and even more importantly, the BibTeX meta-data needs to be entered only once.

[1] Also known as Microsoft Office or MSOffice

Test collection Once a large body of meta-data and PDF files of publications is available, this body can serve as a test collection for information retrieval experiments. On the short term, the PRONIR project may benefit from such a collection.

Show case Prototypes needed for the above experiments can be integrated into the *IRIS PMS*, both as a showcase of our results in this area and as a usefull tool to search our own *IRIS PMS*.

1.2 Key design principles

In order to achieve these potential benefits, the following key design principles will be adhered to:

Enter once Publication data, in particular pertaining to our own publications, should be entered once only. This covers administration for IRIS purposes, ICIS/University purposes, as well as BibTeX/referencing purposes.

One source All data/files should be available from *one source* (i.e. database/website). This includes BibTeX data, ppisa, sources (LaTeX and/or Office formats), pdf files, etc.

Generic functionality Some of the functionality needed from the *IRIS PMS* will be IRIS specific. However, a lot of it will be functionality that is useful to any research community. Making this separation in the system would allow us to turn this project into an Open Source project with the potential of letting other groups in the community help in the realization of the project.

An initial separation can be made between:

1. Dealing with publication data in general (storage, retrieval, searching, quality checks).
2. Maintaining lists of publications per organizational unit or project. E.g. PRONIR, IRIS, SoS, ICIS, etc.
3. Tracking of publications along their life-cycle (e.g. the PPisa & Tech report system).

On-demand migration & import We currently already have a large database of bibtex entries, some of which are even obsolete. To clean up this collection, and to guard the quality of the associated meta-data (e.g.
BibTeX details), we propose to use an on-demand migration & import strategy.

What we mean by this is that users can use BibTeXkeys in their publications which refer to entries that do not *yet* exist in the new *IRIS PMS* database, but which do exist in either the old IRIS database, or one of the other publication databases which we can interface with. Whenever a citation is made to such a publication, the pre-existing publication should be migrated over to the new *IRIS PMS* (this process will need to include a 'by hand' validation check).

At first the on-demand import will be limited to the migration of entries from the existing database. Future versions of the *IRIS PMS* will be able to import from other sources as well.

2 Requirements

This section contains an informal discussion of the system's usage in terms of *use cases*. Here you will find the use cases that have been elicited out of the various discussions with the future users of the *IRIS PMS*.

2.1 Problem Statement

To implement the *IRIS PMS* as described in the introduction, providing functionality that will ultimately deliver the previously described benefits.

2.2 Statement of work

The realization of the *IRIS PMS* will be considered completed when each and every use case has been implemented. An analysis of the requirements will be made using use cases, which will function as the contract with which we determine completion of the broker component.

2.3 Stakeholders

The following have been identified as stakeholders in this project:

- Theo van der Weide - department manager, lead researcher for IRIS.
- Erik Proper - lead researcher for IRIS, primary user testing.

2.4 Actors

The following list includes all actors that are the initiation point for a use case:

- User (internal IRIS researcher).
- External User (browsers of the system).
- Administrator (responsible for system itself).
- Processor (processes submitted publications).

2.5 Defined use cases

The following table shows a listing of use cases as defined for completing the *IRIS PMS* functionality:

- Add publication
- Update publication details
- Import publications from external system
- Generate publications list
- Generate BiBTeX file
- Publication lifecycle tracking
- Request publication data
- Quality checks

2.5.1 Add publication

This involves entering technical reports of the *IRIS* research group into the *IRIS PMS*, as well as triggering, tracking and monitoring of the workflow needed to producing the technical report version, and consequent submission to a journal/conference and possible publication. Furthermore, we wish to allow addition of external publication that circumvent the requesting of institutional technical report numbers. These will be expanded into extra scenarios.

Use Case Name:	**Add publication**
Description:	A user can submit a publication for addition to the PMS. This will also facilitate the process of receiving a *PPISA number* and when applicable, a *Tech Report number*.
Actors:	User
Preconditions:	1. PMS must be available for user.
Triggers:	User submits a new publication to the PMS user interface (document, BiBTeX entry, submission data and an abstract).
Basic Course of Events:	1. User calls up the PMS submission URL. 2. User fills in the online form. 3. User submits form. 4. User asked to upload PDF of submitted document. 5. User uploads PDF. 6. Notification is sent via email to processing administrator. 7. Link provided to print copy of PDF for processing administrator. 8. Data is added to PMS database.
Exceptions:	1. Incorrect data gives error with back button. 2. File upload failure gives error with back button.
Postconditions:	1. Publication has been submitted. 2. Publication is available in PMS (status == submitted). 3. Publication data has been e-mailed to submission administrator. 4. Publication has been printed for submission administrator.

2.5.2 Update publication details

Look-up an entry and change its data, sources, PDF or abstract.

Use Case Name:	**Update publication details**
Description:	A user can submit changes to an existing PMS publication.
Actors:	User
Preconditions:	1. PMS must be available for user. 2. Publication exists in PMS.
Triggers:	User initiates process of updating an existing PMS publication.
Basic Course of Events:	1. User calls up the PMS update URL. 2. User selects single publication from list. 3. User can choose to edit publication details. 4. User can choose to edit PDF (submit new PDF). 5. User can choose to edit sources (submit new sources). 6. User submits edited details, new PDF or new sources. 7. Data is modified in PMS databases.
Exceptions:	1. PMS database is unavailable, gives error message and back button. 2. Submitted incorrect data, gives error message and back button.
Postconditions:	1. Publication has been updated. 2. Publication is available in PMS system (status == updated)

2.5.3 Import publications (from external systems)

This involves publications already entered in the older prototype system and it involves the entering of publication data (e.g. citation information, abstract, PDF file, and sources) from the IRIS research group (or any other group) into the *IRIS PMS*. This deals with publications that already have been *published* as an article, technical report, etc. In other words, these publications do *not* need to be converted into a technical report anymore.

Use Case Name:	**Import publications**
Description:	A user can import a publication from an external source. This can be from the old publications system or for any publication that does not need to receive *PPISA numbers* or *Tech Report numbers*.
Actors:	User
Preconditions:	1. PMS system must be available for user. 2. Publication to be imported must be available to PMS. 3. Import files in old publication system format must be available.
Triggers:	User requests process of importing an external publication.
Basic Course of Events:	1. User calls up the PMS import publication URL. 2. User provides bibtex file and submits to import form. 3. Form presented to User with submitted data inserted where possible. 4. User fills in the online form and/or edits imported data. 5. User submits form, supplies *addpub use case* information. 6. PMS imports external publication.
Exceptions:	1. Publication data submitted in file is not parsable for inclusion into form, just present a blank form for import. 2. Submission is rejected for any reason, message user and provide back buttion with history.
Postconditions:	1. Publication has been added to PMS database. 2. Publication is available in PMS system.

2.5.4 Generate publications list

This should be able to generate pages for personal CV's, IRIS website, etc.

Use Case Name:	**Generate publications list**
Description:	Provides a mechanism to generate listings of publications based on a given group or person. It should be possible to retrieve pre-processed, ready to go HTML pages of the results or an ASCII bibtex file listing of the results for further user modification.
Actors:	User
Preconditions:	1. PMS system must be available for user.
Triggers:	User submits a publication list request.
Basic Course of Events:	1. User can request a publications listing based on: (a) last name (b) first name (c) group (d) year 2. User receives HTML or ASCII bibtex file with listing.
Exceptions:	None.
Postconditions:	1. Publications listing in HTML has been generated. 2. Publications listing in ASCII bibtex has been generated.

2.5.5 Generate BiBTeX file

Need to generate BiBTeX file based on a complete *IRIS PMS* listing, a listing from AUX as input or a single entry from an alias/key.

Use Case Name:	**Generate BiBTeX file**
Description:	A user can generate a BiBTeX file based on a supplied AUX file, request a complete BiBTeX of the PMS or request a single BiBTeX entry based on a key/alias list. The user can also request a listing of key/aliases.
Actors:	User
Preconditions:	1. PMS system must be available for user. 2. User supplies valid AUX file.
Triggers:	User submits either a valid AUX file, a request for all BiBTeX entries in the PMS or a valid key/alias from a provided list.
Basic Course of Events:	1. User supplies an AUX file. 2. User supplies a publication id. 3. User requests all BiBTeX entries in PMS. 4. User receives a listing of BiBTeX entries.
Exceptions:	PMS database unreachable, reports error.
Postconditions:	1. User has a BiBTeX file with requested entries.

2.5.6 Publication life-cycle tracking

The status will be available for user viewing to show the current processing of a publication. Also able to see the past history of the publication.

Use Case Name:	**Publication lifetime tracking**
Description:	User can request a publications history.
Actors:	User
Preconditions:	1. PMS is available.
Triggers:	User requests the history of a publication.
Basic Course of Events:	1. User requests the history of a publication based on provided list. 2. Publication history presented.
Exceptions:	PMS database unavailable, reports error.
Postconditions:	1. User has been presented with the requested publications history.

2.5.7 Request publication data

Search in *IRIS PMS* for publications based on title / abstract keyword search criteria.

Use Case Name:	**Request publication data**
Description:	Provides a mechanism to search for a single publication or groups of publications based on title / abstract keyword searching.
Actors:	User
Preconditions:	1. PMS is available.
Triggers:	User submits a search request to PMS.
Basic Course of Events:	1. User submits a keywords search string. 2. PMS validates and searches titles and/or abstracts for given string. 3. User is presented with a list of publications that match search string.
Exceptions:	PMS database is unavailable, reports error.
Postconditions:	1. User presented with list of publications based on search criteria.

2.5.8 Quality checks

Merge two (or more) entries. Show all fields and let user decide which ones to use, including the option to change things all together.

- People (names)
- Names of publishers
- Addresses of publishers
- City names
- Being part of a conference proceedings

Use Case Name:	**Quality checks**
Description:	Provides mechanism to search for possible duplicate entries in PMS. Provides tools and process to resolve duplicate entries (merge, diff, edit, save, choose).
Actors:	Administrator
Preconditions:	1. PMS is available.
Triggers:	Administrator initiates a quality check.
Basic Course of Events:	1. Admin initiates a quality check. 2. Admin is presented with a list of possible duplicates. 3. Admin can view the entries. 4. Admin can edit entries. 5. Admin can delete entries. 6. Admin can merge entries. 7. Admin can mail an entry to IRIS member for consultation.
Exceptions:	PMS database is not available.
Postconditions:	1. Duplicate entries are no longer found in PMS.

2.6 Scenarios

Each use case from the previous section are further expanded into completed scenarios here. These are the filled instances of each use case, containing step by step actions and exact data elements to be used by each use case.

2.6.1 Scenario: Add publication - request techreport number

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - request techreport number**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Submitter name • Title • Authors • Keywords • Abstract • submitted to publication-url • submitted date • file • Note • Annote • Month • Year • Institution • Address • Research group 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database. 6. Processing administrator receives notification via email that a submission is awaiting processing.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.2 Scenario: Add publication - article

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - article**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • Journal • Year • PDF file • Month • Volume • Number • Pages • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.3 Scenario: Add publication - book

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - book**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author or Editor • Title • Publisher • Year • PDF file • Month • Volume • Number • Series • Edition • Address • Research group • Translated from resource • Annote • Note • Isbn 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.4 Scenario: Add publication - booklet

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - booklet**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Title • PDF file • Author • Howpublished • Month • Year • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.5 Scenario: Add publication - inbook

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - inbook**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author or Editor • Title • Chapter and/or Pages • Publisher • Year • PDF file • Book id (reference to book) • Month • Volume • Number • Series • Edition • Type • Address • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.6 Scenario: Add publication - incollection

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - incollection**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • Publisher • Year • PDF file • Book id (reference to book) • Editor • Month • Volume • Number • Series • Edition • Type • Chapter • Pages • Address • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.7 Scenario: Add publication - inproceedings

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - inproceedings**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • Booktitle • Year • PDF file • Proceedings id (reference to proceedings) • Editor • Month • Volume • Number • Series • Pages • Publisher • Organization • Address • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.8 Scenario: Add publication - manual

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - manual**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Title • PDF file • Author • Organization • Address • Edition • Month • Year • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.9 Scenario: Add publication - mastersthesis

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - mastersthesis**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • School • Year • PDF file • Month • Type • Address • Research group • Translated from resource • Annote • Note • Isbn 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.10 Scenario: Add publication - misc

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - misc**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • PDF file • Author • Title • Howpublished • Month • Year • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.11 Scenario: Add publication - phdthesis

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - phdthesis**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • School • Year • PDF file • Month • Type • Address • Research group • Translated from resource • Annote • Note • Isbn 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.12 Scenario: Add publication - proceedings

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - proceedings**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Title • Year • PDF file • Editor • Month • Volume • Number • Series • Publisher • Organization • Address • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.13 Scenario: Add publication - techreport

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - techreport**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • Institution • Year • PDF file • Type • Month • Number • Address • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.14 Scenario: Add publication - unpublished

The following details an example usage of the use case including relevant data.

Use Case Name:	**Add publication - unpublished**
Use Case Steps:	1. User sends a request to PMS to add a publication. 2. The request is verified to contain the required data (underlined data is required): • Author • Title • PDF file • Month • Year • Research group • Translated from resource • Annote • Note 3. User receives verification that data has been accepted. 4. Data is added into PMS databases. 5. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data. 2. User receives rejected file upload (with back button).

2.6.15 Scenario: Update publication details

The following details an example usage of the use case including relevant data.

Use Case Name:	**Update publication details**
Use Case Steps:	1. User submits search string for search based on publication titles. 2. User chooses a single publication to edit from the presented list. 3. User presented with publication data for editing. 4. User presented with possibility to upload a new PDF. 5. User presented with possibility to upload new sources. 6. User presented with publication details for updating. 7. User submits changes. 8. Data is modified in PMS database. 9. History entry is made for this publication.
Alternative Path:	1. User search string results in no publication, message and new search presented. 2. User receives rejected update, for any reason, message and back button provided.

2.6.16 Scenario: Import publications

The following details an example usage of the use case including relevant data.

Use Case Name:	**Import publications**
Use Case Steps:	1. User sends a request to PMS to import an external publication. 2. User can load a bibtex file to the import form. 3. Import form will be filled with submitted file data as was parsable. 4. User edits and/or fills in the import form. 5. The request is verified to contain the required data based on bibtex type: (a) BiBTeX data based on type (b) abstract (c) file (PDF) (d) source (ZIP) 6. Data is uploaded into PMS. 7. History entry added to database.
Alternative Path:	1. User receive reject message (with back button) due to invalid data.

2.6.17 Scenario: Generate list of publication

The following details an example usage of the use case including relevant data.

Use Case Name:	**Generate publications list**
Use Case Steps:	1. User requests a publications list generation. 2. The request is verified to contain the required data: • author last name • author first name • group name • year 3. User provided with requested publications list in requested format.
Alternative Path:	1. User request results in empty listing, empty list is shown.

2.6.18 Scenario: Generate BiBTeX file

The following details an example usage of the use case including relevant data.

Use Case Name:	**Generate BiBTeX file**
Use Case Steps:	1. User submits a generate bibtex entry request. 2. The request is verified to contain one of the following: • AUX file • key • alias • request all entries 3. User receives list of requested results.
Alternative Path:	1. User request results in an empty listing, send warning with back button.

2.6.19 Scenario: Publication life-cycle tracking

The following details an example usage of the use case including relevant data.

Use Case Name:	**Publication lifetime tracking**
Use Case Steps:	1. User presented with a list of all publications (key - title - authors). 2. User selects publication from a list to view. 3. The history as known to PMS is displayed for the selected publication.
Alternative Path:	1. User request results in empty history, display empty history.

2.6.20 Scenario: Request publication data

The following details an example usage of the use case including relevant data.

Use Case Name:	**Request publication data**
Use Case Steps:	1. User submits a search string request. 2. User indicates wish to search title and/or abstracts. 3. Matching publication data is presented to the user for viewing only.
Alternative Path:	1. User request fails, error message shown and back button.

2.6.21 Scenario: Quality checks

The following details an example usage of the use case including relevant data.

Use Case Name:	**Quality checks**
Use Case Steps:	1. Admin requests a quality check be run on system. 2. PMS searches for possible duplicates in database. 3. PMS presents list of possible duplicates for inspection. 4. Admin can view entries. 5. Admin can edit entries. 6. Admin can delete entries. 7. Admin can merge two entries. 8. Admin can mail entry to IRIS member for consultation.
Alternative Path:	1. None.

3 Domain Model

In this section we detail our domain with a conceptual model using an *ER* diagram.
Due to the rather large size of the model it has been placed online for viewing at:
`http://osiris.cs.kun.nl/iriswp/projects/ER-Resource.png`

4 Database schema

Our database schema has been left as a generic table listing here so as to be database independent. Field lengths and data types are also left as an exercise to the user as this is a matter of taste in the implementation.

4.1 Tables

The following tables have been generated from the ER model. The table names are in boldface and the keys underlined. Foreign keys are shown with (FK) abbreviation:

RESOURCE (id, isbn, annotate, month, year, note, class, type, howpublished, resource key, title, research group, source, abstract, content, translated from resource)

URI (resource id (FK), url, last check)

HISTORY (resource id, date, status)

ACTOR (id, first name, last name, suffix, email, region long, region short, city long, city short, nation)

ACTORTYPE (actor type, resource id (FK), actor id (FK), order number)

WEBSITE (resource id (FK))

MISC (resource id (FK))

UNPUBLISHED (resource id (FK))

BOOKLET (resource id (FK))

TECHREPORT (resource id (FK), tech number, typereport, institution)

INCOLLECTION (resource id (FK), book id, book number, book volume, book edition, book series, book title, chapter, pages)

INPROCEEDINGS (resource id (FK), proceedings id, book number, book volume, book series, book organization, booktitle, pages)

PROCEEDINGS (resource id (FK), volume, series, organization, number)

MSC (resource id (FK), school)

PHD (resource id (FK), school)

ARTICLE (resource id (FK), issue number, issue volume, journal title, pages)

BOOK (resource id (FK), number, volume, edition, series)

INBOOK (resource id (FK), book id, book number, book volume, book series, chapter, pages)

MANUAL (resource id (FK), edition, organization)

ALIASES (resource aliases, resource id (FK))

CODE (resource code, resource id (FK))

INSUBMISSION (resource id (FK), submitter, submission, submitdate, keywords, date)

5 Design

The *IRIS PMS* design will be detailed in the following section. First a design overview will be shown with each entry point to the *IRIS PMS* system detailed. An overview of the classes involved will be only briefly outlined with the details of the current running implementation.

5.1 Overview

The following is a brief overview of the classes involved in the *IRIS PMS*, see Fig 1.

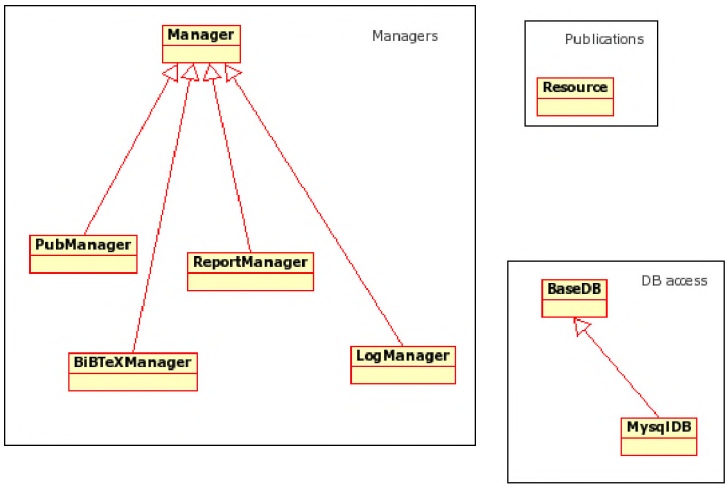

Figure 1: IRIS PMS class diagram overview

This overview does not include the current state of the various classes as these are viewable online at the *IRIS Scientific Programmers* website `http://www.cs.ru.nl/iriswp`. There you will find thee details of each classes interface with browsable documentation.

5.2 Manager class

The *manager class* is an abstract class that needs to be implemented by the other managing classes below. It is used to keep track of the type of manager it is and to store any error message that may be generated during its lifetime.

5.3 Log manager class

The *log manager class* implements the *manager class* to provide our *IRIS PMS* with a varied interface for the logging of its actions during usage. This class offers three logging services:

- file - logging to a given file.
- syslogger - logging through system logging facilities.
- html - logging to standard output using HTML markup.

5.4 Publication manager class

This is the central point to access the *IRIS PMS* services. This class provides the starting point to add, remove or change any entries in the *IRIS PMS* database. Furthermore, it also provides services that can be used for reporting in the various formats offered by the *report manager class*. This class makes extensive usage of the rest of the *IRIS PMS* framework classes to provide its rather extensive service list.

It is most likely that future versions of the *IRIS PMS* framework will see this class being divided into sub-classes. This will become a necessity as the services offered by the *publication manager class* grow with the *IRIS PMS* over time.

5.5 Report manager class

A focus on the reporting of *IRIS PMS* contents is the primary job of the *report manager class*. This focus for the 1.0 version is to be on *HTML* and *ASCII* formats.

The *report manager class* provides users with HTML markup pages of their personal publications, overviews based on groups and/or specific years and just about any type of report that can be imagined. These reports are based on the current data in the *IRIS PMS* and can be produced in almost any form desired. Currently we have provided HTML browsable forms and an ASCII text bibtex format. More can be expected over time.

5.6 BiBTeX manager class

The *bibtex manager class* is a very basic parsing tool used to import from the pre-existing IRIS systems bibtex formats. This is not a complete bibtex parser yet. It also provides a service to take normal text input strings and generate correct bibtex entries, applying all necessary formatting to achieve usable bibtex entries.

5.7 Resource class

To be able to manipulate a single resource, this simple class was created. It is a tool for loading a single resource and being able to obtain its various data elements. It consists of mainly *set* and *get* methods that operate on the various elements of a resource.

5.8 Base database class

An abstract class that needs to be implemented for a specific database application. This class provides the template for basic database communications.

5.9 Mysql database class

This class implements the *base database class* for a *Mysql database*. All manner of connections, query execution and disconnecting is taken care of by this class.

6 Data definitions

The various data elements are shown here to clarify their usage and definitions within the PMS.

abstract - *string:* the abstract from the resource.

actor_id - *integer:* foreign key that identifies an actor uniquely (unsigned gives us 4,294,967,296 actors)

actor_type - *string:* a description of the type of actor (such as author, editor, etc).

annotate - *string:* an annotation about the resource.

authority - *string:* - the school or organization that initiated the resource creation (school by a PhD, organization by a book, etc).

booktitle - *string:* title of a book, part of which is being cited. For book entries use the title files (quoted from bibtex definitions).

book_edition - *string:* the book edition which contains the resource.

book_id - *string:* the book resource id in which this incollection or inbook resource was published.

book_number - *integer:* number of the book from the resource.

book_organization - *string:* organization name that sponsors the book containing resource.

book_series - *string:* name of a series or set of books containing the resource.

book_title - *string:* title of book containing the resource (not same as *booktitle.*

book_volume - *integer:* book volume that contains the resource.

chapter - *integer:* chapter which contains the resource.

city_long - *string:* long name of the city (such as New York).

city_short - *string:* short name of the city (such as NY).

class - *string:* the current publication type (book, chapter, tech report, etc).

code - *string:* (multi-valued field) name+number codes used to reference the resource.

context - *binary:* the PDF file of the resource.

date - *string:* a date stamp in the form YYYY–MM–DD.

email - *string:* email address for the actor.

first_name - *string:* first name of actor.

howpublished - *string:* how something strange has been published.

keywords - *string:* the submitted technical reports keywords.

id - *string* | *integer:* the identifying key for a resource or actor.

institution - *string:* name of organization sponsoring the technical report.

isbn - *string:* the registered ISBN number.

issue_number - *integer:* issue number of journal containing resource.

issue_volume - *integer:* issue volume of journal containing resource.

journal_title - *string:* title of journal containing resource.

last_check - *string:* date this url was last verified.

last_name - *string:* last name of actor.

middle_name - *string:* middle name(s) of actor.

month - *integer:* the month the resource was published (unpublished then month resource was written).

nation - *string:* name of the nation.

note - *string:* any extra information about the resource.

number - *integer:* number of journal, magazine, technical report or work in a series (taken from

bibtex definitions).

organization - *string:* organization name that sponsors the conference or publishes manual containing resource.

pages - *string:* the page or range of numbers containing the resource.

ppisa - *integer:* PPISA number assigned to the resource.

proceedings_id - *string:* the proceedings resource id in which this inproceedings resource was published.

publisher - *string:* publisher of the resource.

region_long - *string:* long name of the region (such as California).

region_short - *string:* short name of the region (such as CA).

resource_aliases - *string:* (multi-valued field) other names for the resource.

research_group - *string:* owning group of the resource (department, organization, etc.)

resource_author - *string:* (multi-valued field) author of the resource.

resource_editor - *string:* (multi-valued field) editor of the resource.

resource_id - *string:* the identifying foreign key for a resource.

resource_key - *string:* used for alphabetizing, cross-referencing and creating a liable when author and editor infomation is missing.

resource_translator - *string:* (multi-valued field) translator of the resource.

school - *string:* name of school where thesis was written (PhD/Masters).

series - *string:* name of a series or set of books containing resource (take from bibtex definitions).

source - *binary:* the source files of the resource in a single compressed file.

status - *string:* a description of the status that the resource has at that moment.

submission - *string:* where this resource was submitted.

submitdate - *string:* date of submission.

submitter - *string:* name of IRIS member that submitted this resource.

suffix - *string:* any suffix the actor may have in name.

tech_number - *string:* technical report number assigned to resource.

title - *string:* title of the resource.

translated_from_resource - *string:* the original resource from which this resource was translated (contains a resource_id).

type - *string:* the bibtex type (proceedings, book, article, etc.) of this resource.

typereport - *string:* the type of technical report if resource is a technical report.

url - *string:* a unique URI description.

volume - *integer:* volume of journal or multivolume book (taken from bibtex definitions).

year - *integer:* the year the resource was published (unpublished then year resource was written).