Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever

Coma

WS 2004/05

Data Spec. (5)

17. 01. 2005

Termin: 17. 01. 2005

Abstract

This document (at the current stage) descibes the data that are to be represented in the Tool. It is also available via the website. It is not the full specification, in particular, no dynamic behavior is yet represented or described (except by some hints.)

It serves to *consolidate* at least this core part and allow to push ahead, in particular the SQL-group. It is considered *consolidated*, in that is is final up-to necessary changes. To reflect the development, the specification is qualified with a versioning number.

The document does not yet mention yet all the restrictions, the dynamic apsects, or the scenarios and phases and further information that has been found in the specifications generated in the first phase.

The main contributors (apart from input during discussions) concerning the actual representation in SQL have been Sandro Esquivel, Tom Scherzer, Gunnar Biederbeck, and Mohamed Albari.

version: 5 (`$Id: abstract.tex 1266 2005-01-11 08:25:56Z ms $`)
status: transient
previous version: v2

We have chosen an (almost standard-conform) UML-class diagram notation. (cf. Figure 1).

The diagram was worked out after discussion on the bulletin board in the plenum. It took inspiration from the two specification deliverables [ESW04] [MSS04], at least the static part, for instance the ER-diagram from [ESW04], but also from different variations from the data model of this document, version 2.

The graphical representation from Figure 1 is simplified insofar as plain fields are left out. Those are specified in more detail in the SQL-representation. Apart from the information here, some textual file has been produced, accessible via the subversion server under `sql/db_schema.txt`.

1 SQL-model

SQL-Spec

Relevant is only the SQL-Spec.

The table conferences (cf. Listing 1) contains all the conferences being hosted by the server.

Listing 1: DB: conference

```
CREATE TABLE Conference
(
    id                              INT NOT NULL AUTO_INCREMENT, -- Coma = conference service = many conferenc
    name                            VARCHAR(127) NOT NULL,       -- Name/Acronym of the conference
    homepage                        VARCHAR(127),                -- webpage of the conference
    description                     TEXT,                        -- ''Werbung''/Beschreibung fuer die Konfere
    abstract_submission_deadline    DATE,                        -- Abgabe einer Kurzfassung
    paper_submission_deadline       DATE,                        -- Abgabe des Papiers, danach beginnt die Beg
```

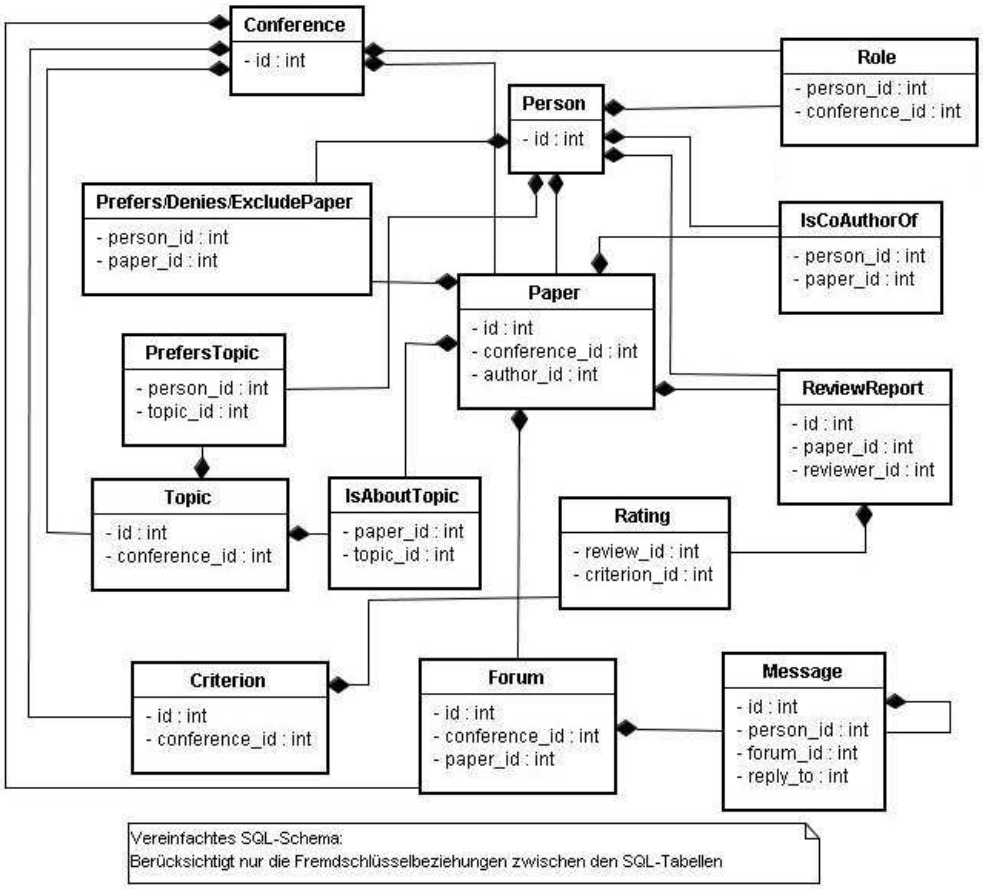

Figure 1: Class diagram

```
    review_deadline             DATE,          -- Abgabe der Bewertung durch die Gutachter
    final_version_deadline      DATE,          -- Endversion des (evntl. revidierten) Papier
    notification                DATE,          -- Benachrichtigung der Autoren (Ja/Nein + K
    conference_start            DATE,          -- Beginn der eigentlichen Konferenz (nicht d
    conference_end              DATE,          -- analog
    min_reviews_per_paper       INT,           -- vorzugsweise: Mindestanzahl Gutachten
    PRIMARY KEY (id)
) TYPE = INNODB;
```

Most of the entries in the table should be self-explanatory. For the deadlines, the following ones are foreseen, in the order of events:

1. The *abstract submission deadline.* This typically is a (secondary) deadline for *authors* shortly before the paper submission deadline. It allows the authors to upload an abstract of their papers. There is not much "semantics" behind this deadline, only that experience shows that it helps in organizing the conference to know in advances how many papers there will be, about what topics etc. A further advantage of the abstract submission deadline is that it may "encourage" the authors to try to meet the real paper deadline a little

2. The *paper submission deadline.* This is the most important deadline for authors, namely: when must they finished their work! After that deadline, no new papers or new versions of the paper may be uploaded.[1]

3. The *review deadline:* That's the first deadline for the *reviewers,* namely: when must they have finished their reading job and must have handed in the grades; on the basis of this information, the discussion and selection starts.

4. The *notification deadline:* this is the deadline when the *selection* has ended and when the authors are notified about their success or failure.

5. The *final version* deadline: afterwards, the successful authors may be required to up-load the "very final version" to be printed, sometimes also called *camera ready version*, which takes into account the criticism of the reviewers.

The *persons* using the tool are all kept in the table `Person`. Again most fields should be self-explanatory. Not all fields of a person must be filled in; in order to facilitate communication, we require the *email address*; for secure identification furthermore a password (cf. Listing 2). An

Listing 2: Person

```
CREATE TABLE Person                                    -- alle natuerlichen Personen
(
    id              INT NOT NULL AUTO_INCREMENT,
    first_name      VARCHAR(127),
    last_name       VARCHAR(127) NOT NULL,
    title           VARCHAR(32),
    affiliation     VARCHAR(127),
    email           VARCHAR(127) UNIQUE NOT NULL, -- UNIQUE wird ignoriert (?)
    phone_number    VARCHAR(20),
    fax_number      VARCHAR(20),
    street          VARCHAR(127),
    postal_code     VARCHAR(20),
    city            VARCHAR(127),
    state           VARCHAR(127),
    country         VARCHAR(127),
    password        VARCHAR(127) NOT NULL,
```

Crucial for the interaction of persons with he tool are the *roles* the persons play (cf 3. Some of them were mentioned informally in the requirements specification [KSS04]. In the course of the semester, we reached at *no agreement* how to represent the role. Thus, neither the representation no the exact choice of roles are indicated in the shown SQL-code.

[1] A exception may be, that the chair make a personal exception or something, but that's only by circumventing the normal process.

Listing 3: Role

```
CREATE TABLE Role
(
    conference_id   INT NOT NULL,
    person_id       INT NOT NULL,
    role_type       INT NOT NULL, -- allowed: 00,02,03,04,05 (not 01)
                                  -- for meaning, see spec.
    state           INT, -- optional
    PRIMARY KEY (conference_id, person_id, role_type),
    INDEX (conference_id),
    INDEX (person_id),
    FOREIGN KEY (conference_id) REFERENCES Conference (id)
        ON DELETE CASCADE,
    FOREIGN KEY (person_id) REFERENCES Person (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

In the textual representation which is agreed upon as supplementary global spec, we fixed the roles of Table 1 to be mandatory. Note that `admin` (corresponding to `01`) is *not* a role within a conference. The numbers indicate the numerical representation.[2]

role	integer	
without role	00	
chair	02	
reviewer	03	
author	04	
participant	05	
admin	01	not a role!

Table 1: roles

Papers are the things that the authors produce and upload via the tool in the process called *submission.* In general, it's some form of document, typcally in postscript or as pdf format; each document must have an *author* and belongs to one particular conference. The person indicated here as author is also called the *corresponding author* which refers to the one from the authors who takes main resposability to interact with the system. The *abstract* of a paper is a short text which summarizes in a few lines the content of the paper.

Listing 4: Paper

```
CREATE TABLE Paper
(
    id              INT NOT NULL AUTO_INCREMENT,
    conference_id   INT NOT NULL,
    author_id       INT NOT NULL,
    title           VARCHAR(127) NOT NULL,
    abstract        TEXT,          -- Kurzfassung der Artikels
    last_edited     DATETIME,
    version         INT,
    filename        VARCHAR(127),
    state           INT NOT NULL,
    mime_type       VARCHAR(127),
    PRIMARY KEY (id),
    INDEX (conference_id),
    INDEX (author_id),
    FOREIGN KEY (conference_id) REFERENCES Conference (id)
        ON DELETE CASCADE,
    FOREIGN KEY (author_id) REFERENCES Person (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

The *state* of a paper is used in the reviewing phase to indicate certain high-level information concerning the result of the discussion. The states agreed upon are shown in Table 2. The

[2]Note further, that unlike as in previous versions, no extra table `Roles` is used, everyone works with the numarical representation.

numerical representation is included for those implementations, that do not want to use an extra table. The states `accepted` and `rejected` should be self-explanatory. Those are the *definite* state, and it's the goal of the reviewing and discussion phase to achieve agreement in the sense that all submissions are either accepted or rejected in the end. The state `conflicting` means that some contradictory judgements have been handed in by the reviewers. A paper with a conflict therefore indicates an increased need for discussion. The conflict is based on the ratings, but the exact conditions as to when the ratings are considered confliciting is a matter of tuning and experimenting in practial circumstances.

state	num. representation
no special state (not being reviewed)	00
being reviewed	01
being reviewed, conflicting	02
accepted	03
rejected	04

Table 2: State

Each paper has at least one author, as a paper does not write itself. A paper may have more than one author, though, and the additional ones are called in the context of *Coma* coauthors.[3] The table `IsCoauthorOf` associates persons and papers (cf. Listing 5). The name-field is to contain the name of the coauthor as string. As the coauthor does not have an active role in connection with the tool, it may suffices to keep the name of the coauthor; this avoids a full "registration" of the coauthor in the tool. Alternatively, the coauthor can be represented as an entry in the table of persons. As a constraint, exactly one of the field `person_id` or the name field may be non-trivially filled in.

Listing 5: Coauthor

```
CREATE TABLE IsCoAuthorOf
(
    person_id   INT,
    paper_id    INT NOT NULL,
    name        VARCHAR(127),
    INDEX (paper_id),
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

As the name indicates, a *topic* is some area of research of interest for a particular conference, and especially of interest for some of the reviewers of the conference, and it may be a topic dealt with in the papers (cf. Listing 6). In general, each conference has a certain number of topics, which are selected when planning the conference. Also the number of topics cannot be predefined, some conferences like to concentrate on 4 chosen topics, others which to cover 20 or more. So a topic is associated with a conference, and field carrying the semantic information is the `name`-field, which contains a (usually not too long) descriptive string.

Listing 6: Scientific topics

```
CREATE TABLE Topic
(
    id              INT NOT NULL AUTO_INCREMENT,
    conference_id   INT NOT NULL,
    name            VARCHAR(127) NOT NULL,
    PRIMARY KEY (id),
    INDEX (conference_id),
    FOREIGN KEY (conference_id) REFERENCES Conference (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

[3] In real life, of course, all are authors, but on the model, one particular author is singled out, and the others, if any, are called coauthors of the paper.

The topic can be used by authors, which categorize their paper into a number of topics given for the conference and which they feel appropriate when submitting. Furthermore, the algorithm which assigns reviewers to papers may make use of this table (cf. [KSS04]). The corresponding associations between papers and topics respectively between topics and reviewers are shown in Listing 7 and 8

Listing 7: Topic of a paper

```
CREATE TABLE IsAboutTopic
(
    paper_id INT NOT NULL,
    topic_id INT NOT NULL,
    PRIMARY KEY (paper_id, topic_id),
    INDEX (paper_id),
    INDEX (topic_id),
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE,
    FOREIGN KEY (topic_id) REFERENCES Topic (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

Listing 8: Preferred topic

```
CREATE TABLE PrefersTopic
(
    person_id  INT NOT NULL,
    topic_id   INT NOT NULL,
    PRIMARY KEY (person_id, topic_id),
    INDEX (person_id),
    INDEX (topic_id),
    FOREIGN KEY (person_id) REFERENCES Person (id)
        ON DELETE CASCADE,
    FOREIGN KEY (topic_id) REFERENCES Topic (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

Listing 9 shows three further associations of persons, in particular reviewers) with papers. The first one expresses a preference of a reviewer for a paper with the same intention as the preference of a reviewer for a topic, namely for use in the distribution of papers to reviewers. The remaining to tables express situation where it is not possible for a reviewer to review a paper. The first one is the situation where the reviewer simply *refuses* to review the paper for some reason, i.e., his unwillingness to do work on that particular paper is so extreme, that the distribution algorithm guarantees that the reviewer is not assigned this paper.[4] The association `ExcludesPaper` is even more severe. Is may be the case that a single person acts in the role of a reviewer and in the role of an author (or coauthor) in the same conference. In this case he *must* not review is paper or take part in the discussion, and even more: he is not even allowed to follow passively the disucssion about his paper (cf. [KSS04]).

Listing 9: Reviewer-paper associations

```
CREATE TABLE PrefersPaper
(
    person_id  INT NOT NULL,
    paper_id   INT NOT NULL,
    PRIMARY KEY (person_id, paper_id),
    INDEX (person_id),
    INDEX (paper_id),
    FOREIGN KEY (person_id) REFERENCES Person (id)
        ON DELETE CASCADE,
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE
) TYPE = INNODB;

CREATE TABLE DeniesPaper
(
    person_id  INT NOT NULL,
    paper_id   INT NOT NULL,
```

[4] A typical reason for "unwillingness" is that the reviewer is a friend or collegue of the author and thinks he cannot judge the paper without bias.

```
    PRIMARY KEY (person_id, paper_id),
    INDEX (person_id),
    INDEX (paper_id),
    FOREIGN KEY (person_id) REFERENCES Person (id)
        ON DELETE CASCADE,
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE
) TYPE = INNODB;

CREATE TABLE ExcludesPaper
(
    person_id   INT NOT NULL,
    paper_id    INT NOT NULL,
    PRIMARY KEY (person_id, paper_id),
    INDEX (person_id),
    INDEX (paper_id),
    FOREIGN KEY (person_id) REFERENCES Person (id)
        ON DELETE CASCADE,
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

A review report is produced by a rewiewer reflecting his opinion about one paper. The review is partly free-form text, where the reviewer expression freely his opinion and gives hints about possible errors or whatever he feels appropriate to remark, but also contains standardized parts such as categorized (numerical) grades. The free-form text is stored in the summary-, the remarks-, and the confidential-field. The confidential text is the part of the review that the author must not see in the end, whereas all other parts of the review except the identity of the reviewer itself, will be passed to the author.

Listing 10: Report

```
CREATE TABLE ReviewReport
(
    id              INT NOT NULL AUTO_INCREMENT,
    paper_id        INT NOT NULL,
    reviewer_id     INT NOT NULL,
    summary         TEXT,
    remarks         TEXT,
    confidential    TEXT,
    PRIMARY KEY (id),
    INDEX (paper_id),
    INDEX (reviewer_id),
    FOREIGN KEY (paper_id) REFERENCES Paper (id)
        ON DELETE CASCADE,
    FOREIGN KEY (reviewer_id) REFERENCES Person (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

The structures for grading of a submission, i.e., the not-so-free-form of the review report, is shown in Listing 11. In a simular way that the organizes of a conference can choose a number of topics that they wish to be treated in the conference, they can choose a rating schema to assure that the "best" papers are selected. This involves a number of categories, the *criteria,* according to which the papers are to be judged. Typical examples are *technical soundness, relevance to the conference, originality/novelty of contribution, writing style* etc. The table `Criterion` represents the possible criteria chosen for a conference, where the field `name` represents the human understandable string characterizing the criterion, and `description` is used for some more explanatory text about the criterion (for instance, whether a small value is better than a high value or other information which helps the reviewer). The criteria are all represented by numerical values, and `max_value` gives the maximal possible value. The `quality_rating` can be used to build a *weighted* mean of all criteria for an overall rating. The default is the ordinary mean value, i.e., the quality rating is 1 for all criteria for the conference.

The rating of a paper (cf. table `Rating`) then contains the actual grading for a given paper, i.e., the numerical values per criterion for the paper according to the review report at hand (the field `grade`). Apart from the numerical value, the reviewer can add some explanatory text or justification for this grade in the field `comment`.

Listing 11: Rating of papers

```
CREATE TABLE Criterion
(
    id              INT NOT NULL AUTO_INCREMENT,
    conference_id   INT NOT NULL,
    name            VARCHAR(127) NOT NULL,
    description     TEXT,
    max_value       INT,
    quality_rating  INT,
    PRIMARY KEY (id),
    INDEX (conference_id),
    FOREIGN KEY (conference_id) REFERENCES Conference (id)
        ON DELETE CASCADE
) TYPE = INNODB;

CREATE TABLE Rating
(
    review_id     INT NOT NULL,
    criterion_id  INT NOT NULL,
    grade         INT NOT NULL,
    comment       TEXT,
    PRIMARY KEY (review_id, criterion_id),
    INDEX (review_id),
    INDEX (criterion_id),
    FOREIGN KEY (review_id) REFERENCES ReviewReport (id)
        ON DELETE CASCADE,
    FOREIGN KEY (criterion_id) REFERENCES Criterion (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

The last two tables are concerned with the *discussion forum* of a conference. The main purpose is to facilitate the exchange of opinions and coming to a consensus among reviewers during the reviewing and selection phase (cf. Listing 12).

Listing 12: Discussion forum

```
CREATE TABLE Forum
(
    id              INT NOT NULL AUTO_INCREMENT,
    conference_id   INT NOT NULL,
    title           VARCHAR(127) NOT NULL,
    forum_type      INT NOT NULL,
    paper_id        INT,
    PRIMARY KEY (id),
    INDEX (conference_id),
    INDEX (forum_type),
    FOREIGN KEY (conference_id) REFERENCES Conference (id)
        ON DELETE CASCADE
) TYPE = INNODB;

CREATE TABLE Message
(
    id          INT NOT NULL AUTO_INCREMENT,
    forum_id    INT,
    reply_to    INT,
    sender_id   INT NOT NULL,
    send_time   DATETIME,
    subject     VARCHAR(127),
    text        TEXT,
    PRIMARY KEY (id),
    INDEX (sender_id),
    FOREIGN KEY (sender_id) REFERENCES Person (id)
        ON DELETE CASCADE
) TYPE = INNODB;
```

References

[ESW04] Sandro Esquivel, Tom Scherzer, and Jan Waller. Coma conference manager: Specification, November 2004.

[KSS04] Marcel Kyas, Gunnar Schaefer, and Martin Steffen. Coma conference manager: Informal requirement specification, November 2004.

[MSS04] Daniel Miesling, Ulrich Schwarz, and Falk Starke. Coma conference manager: Specification, November 2004.

Index