UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DEPARTMENT OF COMPUTER SCIENCE

MASTER'S DEGREE IN COMPUTER SCIENCE

COURSE DOCUMENTATION
DATABASE

ConferenceHub Inc.

Project Documentation

PROFESSOR:
Prof. Antonio Pellicani

STUDENT:
Andrea Porcelli

ACADEMIC YEAR 2025 - 2026

Contents

Introduction

This document is the project report for the Database course at the Department of Computer Science, University of Bari Aldo Moro.

The project is about designing and building a database for a company called ConferenceHub Inc. that manages academic conferences. The document is divided into chapters: requirements analysis, conceptual design (ER), logical design, physical implementation in Oracle SQL, web application, and a bonus chapter on data warehousing.

Box Types Used in This Document

In this document we use colored boxes to separate different types of content. Each color has a specific meaning:

- **Green — Specifications.** Text taken directly from the original requirements.
- **Violet — Assumption / Design Reasoning.** Explains a design choice or an assumption we made.
- **Blue — Info.** Extra context or useful notes.
- **Grey — SQL Code.** SQL code with syntax highlighting.

Here is an example of each box:

Specifications

This is a green Specifications box. It shows text taken from the requirements.

Assumption / Design Reasoning

This is a violet Assumption / Design Reasoning box. It explains a design decision or an assumption.

Info

This is a blue Info box. It gives additional context or clarifications.

> *This is a grey highlight box, used for short formulas or patterns worth remembering.*

Chapter 1

Requirements Analysis

In this chapter we analyze the original specification using the standard 8-phase method. Each step cleans up and organizes the text so we can use it in the next phase.

The eight phases are:

1. **Choose the right level of abstraction**
2. **Standardize sentence structure**
3. **Linearize and split articulated phrases**
4. **Identify homonyms and synonyms**
5. **Make explicit references between terms**
6. **Build a glossary**
7. **Reorganize phrases by keyword**
8. **Separate data specifications from functional requirements**

1.1 Requirements Gathering Output

Below is the original specification text:

Specifications

Each conference organized by the company "ConferenceHub Inc." is identified by an acronym and has an associated name, the venue where it will take place, the URL of the homepage, and an optional set of sponsors who finance the conference. For each sponsor, the following information is known: name, date the funding was provided, and amount. Each conference is assigned a set of organizers who form the program committee. For each organizer, the following is known: name, affiliation, address, phone number, and email. Each organizer may indicate, for each conference in which they participate as a program committee member, one or more research areas denoted by an acronym and a description. Articles submitted to a conference (an article cannot be submitted to more than one conference) are characterized by a sequential number within the conference, a title, and one or more authors for whom the following information must be maintained: name, affiliation, address, phone number, and email. Note that among the authors, one must be designated as the "contact author" who will receive all communications regarding the submitted article. Articles are also divided into different categories: tutorial, short papers, posters, industrial papers, and research papers. For industrial papers, information (name and address) about all partners who contributed to the research presented in the article must be stored. Based on the specified research area, articles are assigned to program committee members (the number of reviewers per article varies from a minimum of two to a maximum of four) who must prepare a review. The review prepared by a reviewer for a particular article includes a score on originality, significance, and quality of the proposed work, as well as a global score and comments that will be sent to the contact author. Based on the collected evaluations, the status of each article may be either "accepted" or "rejected".

Figure 1.1 shows the same text with colors to highlight the main concepts.

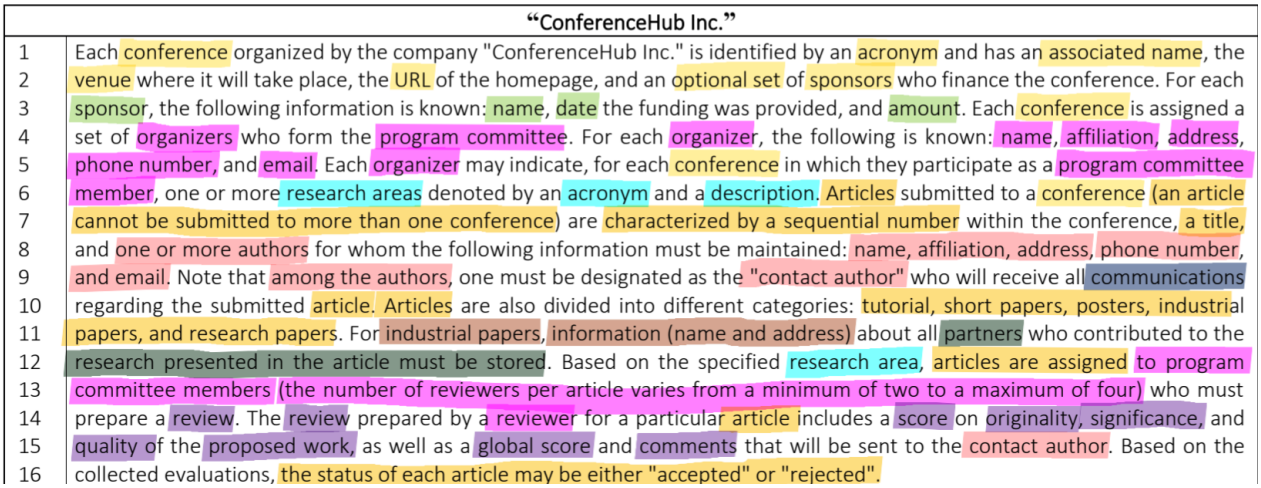

Figure 1.1: Highlighted requirements gathering output.

1.2 Phase 1: Choose the right level of abstraction

We check if any term in the specification is too vague or too specific.

Specifications

Each conference organized by the company "ConferenceHub Inc." is identified by an acronym and has an associated name, the venue where it will take place, the URL of the homepage, and an optional set of sponsors who finance the conference.

No additional work needed.

Specifications

For each sponsor, the following information is known: name, date the funding was provided, and amount.

No additional work needed.

Specifications

Each conference is assigned a set of organizers who form the program committee.

No additional work needed.

Specifications

For each organizer, the following is known: name, affiliation, address, phone number, and email.

No additional work needed.

Specifications

Each organizer may indicate, for each conference in which they participate as a program committee member, one or more research areas denoted by an acronym and a description.

No additional work needed.

Specifications

Articles submitted to a conference (an article cannot be submitted to more than one conference) are characterized by a sequential number within the conference, a title, and one or more authors for whom the following information must be maintained: name, affiliation, address, phone number, and email.

No additional work needed.

Specifications

Note that among the authors, one must be designated as the "contact author" who will receive all communications regarding the submitted article.

Assumption / Design Reasoning

The word "communications" is vague. In this context, the only communication that has real structured data is the review (scores, comments, link to the article). Things like acceptance emails are just notifications and do not need to be stored. So we interpret "communications" as **reviews**.

Specifications

Articles are also divided into different categories: tutorial, short papers, posters, industrial papers, and research papers.

No additional work needed.

Specifications

For industrial papers, information (name and address) about all partners who contributed to the research presented in the article must be stored.

No additional work needed.

Specifications

Based on the specified research area, articles are assigned to program committee members (the number of reviewers per article varies from a minimum of two to a maximum of four) who must prepare a review.

No additional work needed.

Specifications

The review prepared by a reviewer for a particular article includes a score on originality, significance, and quality of the proposed work, as well as a global score and comments that will be sent to the contact author.

Assumption / Design Reasoning

The specification does not say what range the scores have. We assume all scores are **integers from 0 to 10**.

Specifications

Based on the collected evaluations, the status of each article may be either "accepted" or "rejected".

Assumption / Design Reasoning

The specification only mentions `accepted` and `rejected`. But before the reviews are done, the article needs a temporary state. So we add `pending` as a third status. The full set is: {`pending`, `accepted`, `rejected`}.

1.3 Phase 2 – 3: Standardize sentence structure & Linearize phrases

We rewrite all sentences using this standard structure:

Info

For each <subject>, we are interested in <properties>.

Each complex sentence is split into simple, atomic statements.

Info

The following sentences follow the order of appearance in the original specification.

1. For each **conference**, we are interested in its `acronym`, `name`, `venue`, and `homepage URL`.
2. For each **conference**, we are interested in the set of **sponsors** that finance it. The set of sponsors may be empty.
3. For each **sponsor** of a conference, we are interested in its `name`, the `funding date`, and the `funded amount`.
4. For each **conference**, we are interested in the set of **organizers** that form its program committee.
5. For each **organizer**, we are interested in its `code`, `name`, `affiliation`, `address`, `phone number`, and `email`.
6. For each **organizer**, for each **conference** in which they participate as a program committee member, we are interested in the **research areas** they declare. **An organizer may declare one or more research areas per conference.**
7. For each **research area**, we are interested in its `acronym` and `description`.
8. For each **article**, we are interested in its `sequential number` (unique within the conference), its `title`, and its `category`.
9. For each **article**, we are interested in the single **conference** to which it is submitted. An article cannot be submitted to more than one conference.
10. For each **article**, we are interested in the set of **authors** who wrote it. Each article has one or more authors.
11. For each **author**, we are interested in its `code`, `name`, `affiliation`, `address`, `phone number`, and `email`.
12. For each **article**, we are interested in which of its authors is designated as the **contact author**. Exactly one author per article must hold this role.
13. For each **article**, we are interested in its `category`, which is one of: `tutorial`, `short paper`, `poster`, `industrial paper`, `research paper`.
14. For each **industrial paper**, we are interested in the set of **partners** who contributed to the research. Each partner has a `code`, `name` and an `address`.

15. For each **article**, we are interested in the set of **reviewers** (program committee members) assigned to evaluate it, based on research area compatibility. Each article receives between 2 and 4 reviewers.

16. For each **review**, we are interested in its `code`, `date`, `content`, the `originality score`, the `significance score`, the `quality score`, the `global score`, and the `comments`. Scores are integer numbers between 0 and 10. Each review is prepared by exactly one reviewer for exactly one article.

17. For each **article**, we are interested in its `status`: `pending`, `accepted`, or `rejected`.

1.4 Phase 4: Identify homonyms and synonyms

We look for synonyms (different words for the same thing) and homonyms (same word for different things), and we pick one standard name for each concept.

Info

Fixing synonyms now avoids confusion in the next phases and keeps naming consistent.

Table 1.1: Synonyms table.

Terms found in the specification	Explanation	Canonical term
Organizer, Program committee member	Both refer to a person participating in a conference committee.	Organizer
Article, Paper	Both refer to a manuscript submitted to a conference.	Article
Score, Evaluation	Both refer to the numerical ratings assigned during the review process.	Score
Communication, Review	In this context, the only communication worth persisting is the structured review.	Review

No homonyms have been found.

1.5 Phase 5: Making explicit references between terms

Nothing to do here — the sentences from Phase 2–3 are already clear.

1.6 Phase 6: Building a glossary

Table 1.3 lists all the terms we found, with descriptions, synonyms, and connections.

Table 1.3: Glossary of terms.

Term	Description	Synonyms	Connections
Conference	Academic event organized by ConferenceHub Inc., identified by an acronym.	—	Sponsor, Organizer, Article, Research Area
Sponsor	Company or individual financing a conference.	—	Conference
Organizer	Member of the program committee. May also act as a reviewer.	Program committee member	Conference, Research Area, Reviewer
Reviewer	An organizer assigned to evaluate articles.	—	Review, Organizer
Research Area	A domain of study used to match articles with suitable reviewers.	—	Research Area Indication, Article, Conference
Article	Manuscript submitted to exactly one conference, classified by category and evaluated for acceptance.	Paper	Conference, Author, Review, Partner
Author	Person who co-wrote a submitted article. One per article is designated as *contact author*.	—	Article
Review	Evaluation prepared by a reviewer for an article, including scores and comments.	Communication	Article, Reviewer
Partner	Organization contributing to the research of an industrial paper.	—	Article
Research Area Indication	Bridge entity recording the research areas declared by an organizer for a specific conference.	—	Organizer, Conference, Research Area

1.7 Phase 7: Reorganizing phrases by keyword

Now we group the sentences by the main entity they describe.

Conference
For each conference we are interested in an acronym, a name, a location, and a homepage URL. A conference may have zero or more sponsors. Each conference has a program committee composed of at least two organizers.

Sponsor
For each sponsor we are interested in a name. For each sponsorship of a conference, a funding date and a funded amount are recorded. A sponsor may finance one or more conferences.

Organizer
For each organizer we are interested in a code, name, affiliation, address, phone, and email. An organizer may serve on the program committee of multiple conferences. For each organizer–conference membership, the organizer may declare one or more research areas.

Research Area
For each research area we are interested in an acronym and a textual description. Research areas are declared by organizers within a specific conference and are used to match articles with suitable reviewers based on topic compatibility.

Article
For each article we are interested in a sequential number (within its conference) and a title. An article is submitted to exactly one conference, written by one or more authors (exactly one of whom is the contact author), and belongs to exactly one of the five categories. It is assigned to between 2 and 4 reviewers based on research area compatibility and eventually receives a status of `accepted`, `rejected`, or `pending`.

Author
For each author we are interested in a code, name, affiliation, address, phone, and email. An author may write multiple articles.

Review
For each review we are interested in a code, a date, content, scores for originality, significance, quality, global score, and comments (scores are integers between 0 and 10). Each review is prepared by exactly one reviewer for exactly one article. The review and comments are sent to the contact author of that article.

Partner
For each partner we are interested in a code, a name, and an address. Partners are associated only with industrial papers.

Reviewer
A reviewer is an organizer who has been assigned to evaluate articles. Reviewers inherit all attributes of Organizer (code, name, affiliation, address, phone, email). Each reviewer may be assigned to evaluate multiple articles via the Review Assignment relationship.

1.8 Phase 8: Separate data specifications from functional requirements

Now we separate the data requirements (already documented above) from the operations the system must support.

Info

These operations (OP1–OP4) are used in Chapter 3 for the workload analysis and in Chapter 4 as stored procedures.

Table 1.14: Functional requirements: operations and estimated frequencies.

ID	Operation	Type	Frequency
OP1	Insert a new article submission to a conference, including title, category, and the list of authors with their contact information.	I	50/day
OP2	For each program committee member, print the list of articles assigned for review, including article title, category, and the names of all authors.	I	40/day
OP3	Insert a new review for an article, including scores for originality, significance, quality, global score, and comments.	I	80/day
OP4	For each conference, print the list of all accepted articles grouped by category, including title, contact author name and email, and average global score.	I	10/day

Chapter 2

Conceptual Design

In this chapter we build the conceptual schema using the Entity-Relationship (ER) model. We start with a skeleton that shows only the main entities and their connections, then we add attributes, cardinalities, and generalizations step by step.

2.1 Design Strategy

We use a hybrid approach: first we draw the skeleton, then we add all the details (attributes, constraints) to get the complete ER diagram.

2.2 Skeleton of the ER Schema

Before adding attributes, we drew a skeleton that shows only the entities and their relationships. Figure 2.1 is our starting point.

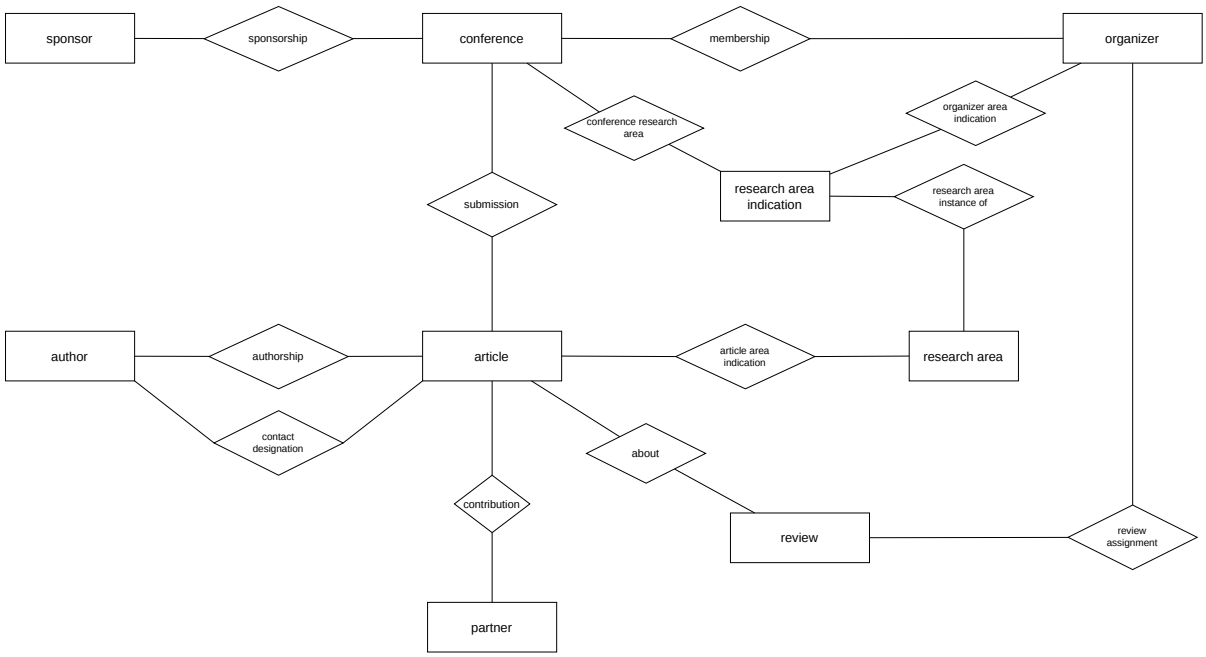

Figure 2.1: Skeleton Entity-Relationship diagram.

2.2.1 Reading the Skeleton

The skeleton has four groups of entities. Below we explain each group.

Conference. Conference is the central entity. Everything connects to it. Sponsors fund conferences through the Sponsorship relationship (many-to-many, a conference can have zero sponsors). Organizers form the program committee through the Membership relationship (each conference needs at least 2 organizers).

Assumption / Design Reasoning

Conference is the central entity because articles, reviews, research areas, and organizers all exist in the context of a conference.

Research Areas and the Indication Entity. This part needs attention because the specification is ambiguous. Let us look at the text:

Specifications

"Each organizer may indicate, for each conference in which they participate as a program committee member, one or more research areas denoted by an acronym and a description."

There is also a second related sentence:

Specifications

"Based on the specified research area, articles are assigned to program committee members [...] who must prepare a review."

These two sentences create an ambiguity. The first says organizers *indicate* research areas. The second says articles are assigned to reviewers *based on* research areas. So research areas are used in two ways: organizers declare them, and they are used to match articles with reviewers.

The key word is "indicate". We interpret it as: when an organizer indicates a research area for a conference, they are defining which topics the conference covers. It is an act of categorization, not just a preference.

This is why Research Area Indication is a separate entity. Each indication connects three things:

- a specific **Organizer** (who makes the indication),
- a specific **Conference** (the context in which the indication happens),
- a specific **Research Area** (the general domain being indicated).

Assumption / Design Reasoning

An organizer in the committee must indicate at least one area. Without areas, there would be no topics and no way to assign articles for review. The indication is what gives structure to the conference.

Without this bridge entity, we would have a ternary relationship between Organizer, Conference, and Research Area. Ternary relationships are hard to read and hard to turn into tables. The bridge entity breaks the three-way link into three binary relationships: *Organizer Area Indication*, *Conference Research Area*, and *Research Area Instance Of*.

Assumption / Design Reasoning

One could argue that we should reify the Membership relationship itself (make it a full entity with the organizer–conference pair, and attach research areas to it). The author of the specification probably had this in mind. But we chose not to do that because: (1) it would make the ER diagram harder to read, and (2) the specification does not explicitly ask for a complex membership structure. We prefer to keep things simple.

Articles, Authors, and Partners. An Article is submitted to exactly one Conference. It has one or more Authors, and one of them is the contact author (who receives review feedback). Articles belong to one of five categories: Tutorial, Short Paper, Poster, Industrial Paper, Research Paper. Only Industrial Papers can have Partners.

Assumption / Design Reasoning

We model the contact author as a separate relationship (Contact Designation) instead of a flag on Authorship. This way the constraint "exactly one contact per article" is visible in the ER diagram.

Reviews and Reviewers. A Review is the evaluation that a reviewer writes for an article. It contains scores and comments. Reviewer is a partial specialization of Organizer: every reviewer is an organizer, but not every organizer reviews. The reviewer role is determined by the Review relationship — no flag is needed.

Assumption / Design Reasoning

The specification mentions "communications" to the contact author. But the only communication worth storing is the Review (with scores and comments). Emails and notifications do not need to be in the database. So we use the Review entity for this.

2.2.2 From the Skeleton to the Full Diagram

After the skeleton, we added primary keys, attributes, and relationship properties to get the complete ER diagram (Figure 2.2).

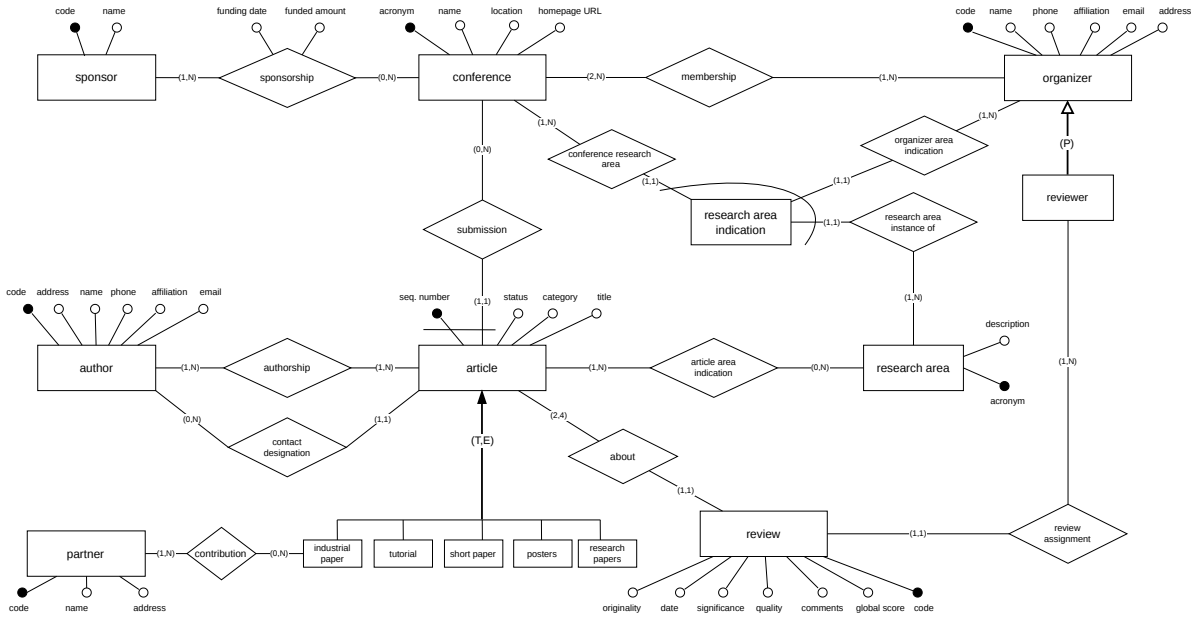

Figure 2.2: Complete Entity-Relationship diagram for ConferenceHub Inc.

2.3 List of Components in the ER

Here we list all entities and relationships in the ER diagram.

2.3.1 Entities

Table 2.1 lists each entity with its attributes and primary key.

Table 2.1: Entities and their attributes.

Entity	Attributes	Identifier (PK)
Conference	acronym, name, location, homepage URL	acronym
Sponsor	name	name
Organizer	code, name, affiliation, address, phone, email	code
Reviewer	*(inherits from Organizer)*	code (inherited)
Research Area Indication	(no own attributes; identified by composite FK)	(composite from FKs)
Research Area	acronym, description	acronym
Article	seq. number, title, category, status, avg_global_score	(conference, seq. number)
Author	code, name, affiliation, address, phone, email	code
Review	code, date, content, originality, significance, quality, global score, comments	code
Partner	code, name, address	code

Info

The entity Reviewer is a *partial specialization* of Organizer: not every organizer reviews articles, but every reviewer is an organizer. The reviewer role is determined by the existence of a Review relationship linking the organizer to an article.

2.3.2 Relationships

Table 2.3 lists each relationship with its participating entities and cardinalities.

Table 2.3: Relationships, participating entities, and cardinalities.

Relationship	Entity 1	Entity 2	Card. E1	Card. E2	Attrs.
Sponsorship	Sponsor	Conference	(1,N)	(0,N)	funding date, funded amt.
Membership	Organizer	Conference	(1,N)	(2,N)	—
Org. Area Indication	Organizer	Res. Area Ind.	(1,N)	(1,1)	—
Conf. Research Area	Conference	Res. Area Ind.	(1,N)	(1,1)	—
Res. Area Instance Of	Res. Area Ind.	Research Area	(1,1)	(1,N)	—
Article Area Indication	Article	Research Area	(1,N)	(0,N)	—
Submission	Article	Conference	(1,1)	(0,N)	—
Authorship	Author	Article	(1,N)	(1,N)	—
Contact Designation	Author	Article	(0,N)	(1,1)	—
About	Review	Article	(1,1)	(2,4)	—
Review Assignment	Review	Reviewer	(1,1)	(1,N)	—
Contribution	Partner	Article	(1,N)	(0,N)	—

Info

The *Research Area Indication* bridge entity breaks the implicit ternary relationship between Organizer, Conference, and Research Area into three binary associations. Each indication records that a specific organizer declared a specific research area within the context of a specific conference.

2.4 Generalizations

Two generalizations have been identified:

- **Reviewer** is a *partial (P)* specialization of **Organizer**.
  - Every reviewer inherits all attributes of Organizer (code, name, affiliation, address, phone, email).
  - Not all organizers are reviewers; only those actually assigned to review articles hold the reviewer role.
  - The specialization is linked to the *Review Assignment* relationship.
- **Article** has a *total, exclusive (T,E)* generalization into five subtypes:
  - **Tutorial**, **Short Paper**, **Poster**, **Industrial Paper**, **Research Paper**.
  - Every article belongs to exactly one category (total coverage, exclusive membership).
  - Only Industrial Papers may have associated Partners via the *Contribution* relationship.

2.5 Design Choices

Here we explain the main design decisions.

Assumption / Design Reasoning

The requirements mention "communications" to the contact author. At first this seems like a separate entity, but the only real structured communication is the Review. Emails and notifications are transient. So we use the Review entity for this and keep the schema simple.

Assumption / Design Reasoning

Program committee members can be assigned to review articles. Instead of a separate Reviewer entity, we model Reviewer as a partial specialization of Organizer. Every reviewer is an organizer, but not every organizer reviews. The reviewer role is determined by whether the organizer has reviews assigned — no flag is needed.

Assumption / Design Reasoning

Each organizer can indicate research areas for each conference. This is naturally a three-way relationship. To avoid a ternary relationship, we use the bridge entity Research Area Indication, which splits it into three binary relationships.

Assumption / Design Reasoning

Articles have five categories. Every article belongs to exactly one, so the generalization is total and exclusive (T,E). The Contribution relationship (with Partner) is attached only to Industrial Paper.

Assumption / Design Reasoning

Instead of a boolean flag on Authorship, we use a separate Contact Designation relationship. This makes the "exactly one contact per article" constraint visible in the ER diagram.

Assumption / Design Reasoning

Some entities do not have a natural unique identifier. Names and emails can be duplicated, so we use artificial `code` attributes for Organizer, Author, Partner, and Review. Conference uses its `acronym` as a natural key. Articles use a composite key: (conference, seq_number).

2.6 Business Rules

These are the constraints that the ER diagram alone cannot express. We enforce them with triggers or CHECK constraints.

Info

Rules labeled (T) are enforced via PL/SQL triggers. Rules labeled (C) are enforced via inline `CHECK` constraints or foreign key design.

BR1 Maximum reviewer cardinality (T). Each article can be assigned to at most 4 reviewers.

BR2 Reviewer conference membership (T). A reviewer assigned to an article must be a member of the program committee of the conference to which the article was submitted.

BR3 Research area compatibility (T). A reviewer should only be assigned to articles whose research areas overlap with the areas they declared for that conference.

BR4 Score range (C). All scores (originality, significance, quality, global score) are integers in the range $[0, 10]$.

BR5 Review completion for status change (T). The status of an article can only change from `pending` after all assigned reviews have been submitted (all reviews have a `global_score`).

BR6 Minimum reviews for status change (T). The status of an article cannot change from `pending` without at least 2 completed reviews.

BR7 Single conference submission (C). An article can be submitted to exactly one conference.

BR8 Contact author subset (T). The contact author of an article must also be one of the authors of that article (Contact Designation $\subseteq$ Authorship).

BR9 Minimum committee size (T). Each conference must have at least 2 organizers in its program committee.

BR10 Positive sponsorship amount (C). The funded amount in a sponsorship record must be strictly positive.

BR11 Acceptance score threshold (T). An article can only be set to `accepted` if its `avg_global_score` ≥ 5.

BR12 Industrial paper partner exclusivity (T). Only articles with category `Industrial Paper` can have partner contributions.

BR13 Average score redundancy maintenance (T). The redundant attribute `avg_global_score` in Article is automatically recalculated whenever a review's `global_score` is inserted, updated, or deleted.

Chapter 3

Logical Design

In this chapter we analyze the workload of each operation and build the logical schema. We also check if storing a redundant attribute is worth the extra cost.

3.1 Relationship and entity workload

Info

The volume table below shows the expected data sizes. The population script in Chapter 4 uses these same numbers.

Table 3.1: Table of Volumes

NAME	TYPE	VOLUME	WHY
SPONSOR	ENTITY	20	Distinct companies.
CONFERENCE	ENTITY	100	Estimated base number.
ORGANIZER	ENTITY	400	Estimated number
REVIEWER	ENTITY	300	Some organizers also review
AUTHOR	ENTITY	3600	Estimated number of authors.
ARTICLE	ENTITY	1200	$\sim$ 120 submissions/event.
INDUSTRIAL PAPER	ENTITY	240	20% of all articles.
TUTORIAL	ENTITY	120	10% of all articles.
SHORT PAPER	ENTITY	600	50% of all articles.
POSTERS	ENTITY	120	10% of all articles.
RESEARCH PAPERS	ENTITY	120	10% of all articles.
REVIEW	ENTITY	3600	Exactly 3 reviews/article.
PARTNER	ENTITY	480	Distinct companies.
RESEARCH AREA	ENTITY	20	Scientific domains.
RESEARCH AREA INDICATION	ENTITY	1200	$\sim$ 3 areas per organizer.
SPONSORSHIP	RELATIONSHIP	500	$\sim$ 5 sponsors per conference, 2.5 sponsors per sponsor on average
MEMBERSHIP	RELATIONSHIP	800	$\sim$ 2 conference per organizer on average (8 per conference)
SUBMISSION	RELATIONSHIP	1200	Each article is submitted to 1 conference
AUTHORSHIP	RELATIONSHIP	3600	$\sim$ 3 authors/article.
CONTACT DESIGNATION	RELATIONSHIP	1200	There is only a single contact designation for each article
CONTRIBUTION	RELATIONSHIP	480	2 partners per ind. paper.
ABOUT	RELATIONSHIP	3600	Each review is associated to a single article
REVIEW ASSIGNMENT	RELATIONSHIP	3600	Each review is made by a reviewer
ARTICLE AREA INDICATION	RELATIONSHIP	2400	$\sim$ 2 areas per article.
CONFERENCE RESEARCH AREA	RELATIONSHIP	1200	Each research area indication is associated to a single conference.
ORGANIZER AREA INDICATION	RELATIONSHIP	1200	Each research area indication is associated to a single organizer.
RESEARCH AREA INSTANCE OF	RELATIONSHIP	1200	Each research area indication is associated to a single research area.

3.2 Table of operations

Here we calculate the cost of each operation. A Read access costs 1, a Write access costs 2. The daily cost is:

$$TOTAL\_COST = COST \times FREQUENCY$$

Table 3.3: Operations and frequencies

OPERATION NAME	TYPE	FREQUENCY
OP 1	I	50 t/day
OP 2	I	40 t/day
OP 3	I	80 t/day
OP 4	I	10 t/day

3.2.1 Operation 1

Insert a new article submission to a conference, including title, category, and the list of authors with their contact information.

Info

This operation writes 1 Article, 1 Submission, 3 Authorship rows, and 1 Contact Designation. It reads Conference and Author to validate foreign keys.

NAME	TYPE	ACCESS TYPE	ACCESS
ARTICLE	E	W	1
SUBMISSION	R	W	1
CONFERENCE	E	R	1
AUTHORSHIP	R	W	3
AUTHOR	E	R	3
CONTACT DESIGNATION	R	W	1
AUTHOR	E	R	1

$$COST = 2 + 2 + 1 + 6 + 3 + 2 + 1 = 17$$

$$TOTAL\_COST = 17 \times 50 = 850 \text{ acc/day}$$

3.2.2 Operation 2

For each program committee member, print the list of articles assigned for review, including article title, category, and the names of all authors.

Info

Read-heavy operation. For each of the 200 reviewers, we read their assignments, the articles, and the authors. This leads to a high access count.

NAME	TYPE	ACCESS TYPE	ACCESS
REVIEWER	E	R	200
REVIEW ASSIGNMENT	R	R	3600
REVIEW	E	R	3600
ABOUT	R	R	3600
ARTICLE	E	R	3600
AUTHORSHIP	R	R	10800
AUTHOR	E	R	10800

$$COST = 200 + 3600 + 3600 + 3600 + 3600 + 10800 + 10800 = 36200$$

$$TOTAL\_COST = 36200 \times 40 = 1{,}448{,}000 \text{ acc/day}$$

3.2.3 Operation 3

Insert a new review for an article, including scores for originality, significance, quality, global score, and comments.
Assuming we already know the article and the reviewer.

Info

Simple write: insert one review and link it to the article and reviewer.

NAME	TYPE	ACCESS TYPE	ACCESS
REVIEW	E	W	1
ABOUT	R	W	1
ARTICLE	E	R	1
REVIEW ASSIGNMENT	R	W	1
REVIEWER	E	R	1

$$COST = 2 + 2 + 1 + 2 + 1 = 8$$

$$TOTAL\_COST = 8 \times 80 = 640 \text{ acc/day}$$

3.2.4 Operation 4

For each conference, print the list of all accepted articles grouped by category, including title, contact author name and email, and average global score.

Info

Reads conferences, articles, and for accepted articles the contact author. Without a cached average score, we would also need to read all reviews. The redundancy analysis below evaluates this trade-off.

NAME	TYPE	ACCESS TYPE	ACCESS
CONFERENCE	E	R	100
SUBMISSION	R	R	1200
ARTICLE	E	R	1200
CONTACT DESIGNATION	R	R	600
AUTHOR	E	R	600
ABOUT	R	R	1800
REVIEW	E	R	1800

$$COST = 100 + 1200 + 1200 + 600 + 600 + 1800 + 1800 = 7300$$

$$TOTAL\_COST = 7300 \times 10 = 73{,}000 \text{ acc/day}$$

3.3 Analysis of the redundancies

Info

We compare the daily cost with and without a cached attribute. If the read savings are much bigger than the extra write cost, the redundancy is worth it.

We check if it is better to calculate the average score on the fly during OP4 or to store it as a redundant attribute (`avg_global_score`) in Article.

3.3.1 Without redundancy

The average is calculated at runtime during OP4.

- **OP 4 Cost**: Fetching 3 reviews for each of the 600 accepted articles costs $1800 + 1800 = 3600$ acc/day. Multiplied by 10 daily iterations: 36,000 acc/day just for reading reviews.
- **OP 3 Cost**: Writing a new review costs 8 acc/day. No extra work. Multiplied by 80 iterations: 640 acc/day.

3.3.2 With redundancy

A redundant attribute `avg_global_score` is stored in Article.

- **OP 4 Cost**: The scores are already cached in the article rows. We skip reading ABOUT and REVIEW entirely. Savings: 36,000 acc/day.
- **OP 3 Cost**: Every time a review is added, we must update the parent Article score. This requires reading the 2 remaining sibling reviews to rebuild the average (1 read for ABOUT, 2 reads for REVIEW = cost 3) plus 1 write on ARTICLE (cost 2). Extra overhead: $5 \times 80 = 400$ acc/day.

3.3.3 Conclusion

Read savings on OP4: 36,000 acc/day. Write overhead on OP3: 400 acc/day. Since $36{,}000 \gg 400$, the redundancy is clearly worth it.

3.4 Partitioning and merging

Here we merge the specializations into their parent entities. Reviewer is absorbed into Organizer, and all Article subtypes (Industrial Paper, Tutorial, etc.) are merged into a single Article table with a `category` attribute to tell them apart.

3.5 Choice of main identifier

Many entities do not have a natural unique attribute. Names can be duplicated and emails can change. Here are our choices:

- **Organizer, Author, Partner, Review**: artificial `code` as primary key (e.g., `ORG_1_2`, `A_42`). We use strings instead of numbers to make the data more readable during testing.

- **Conference**: uses its `acronym` as natural key (e.g., "ICSE", "VLDB").
- **Article**: composite key `(conference_acronym, seq_number)`. In the physical schema we also add a surrogate `article_id` to simplify foreign keys.
- **Research Area**: uses `area_acronym` as natural key (e.g., "DB_SYS", "AI").
- **Research Area Indication**: composite key from its three foreign keys: `(organizer_code, conference_acronym, area_acronym)`.
- **Sponsor**: uses `name` as primary key.

3.6 Logic schema

The final schema with the redundancy included:

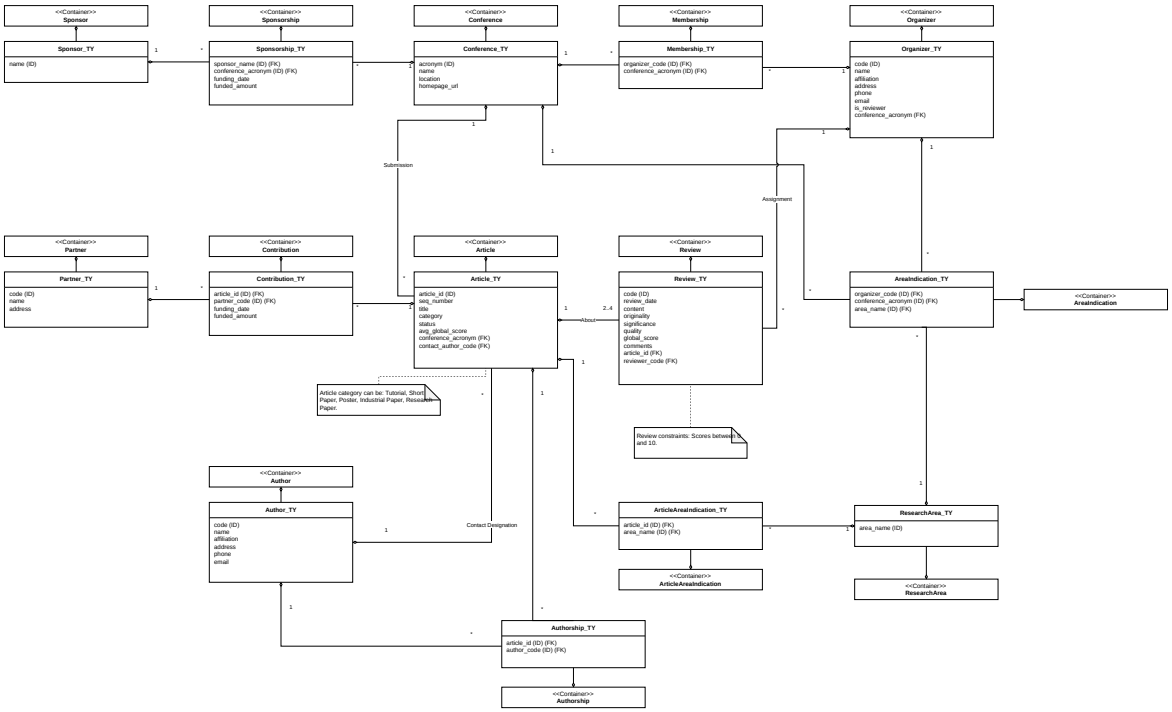

Figure 3.1: UML logical representation defining Object-Relational types.

3.7 Relational Schema

Below is the textual representation of the logical schema, after all transformations.

Info

Notation: underlined attributes form the primary key. *Italic* attributes are foreign keys. Composite primary keys span multiple underlined columns.

Conference(acronym, name, location, homepage_url)

Sponsor(name)

Sponsorship(*sponsor_name*, *conference_acronym*, funding_date, funded_amount)
FK: sponsor_name → Sponsor(name) FK: conference_acronym → Conference(acronym)

Organizer(code, name, affiliation, address, phone, email)

Membership(*organizer_code*, *conference_acronym*)
FK: organizer_code → Organizer(code) FK: conference_acronym → Conference(acronym)

ResearchArea(area_acronym, area_name, description)

AreaIndication(*organizer_code*, *conference_acronym*, *area_acronym*)
FK: organizer_code → Organizer(code) FK: conference_acronym → Conference(acronym)
FK: area_acronym → ResearchArea(area_acronym)
Constraint: (organizer_code, conference_acronym) ∈ Membership

Author(code, name, affiliation, address, phone, email)

Article(article_id, *conference_acronym*, seq_number, title, category, status, avg_global_score, *contact_author_code*)
FK: conference_acronym → Conference(acronym) FK: contact_author_code → Author(code)
Unique: (conference_acronym, seq_number)

Authorship(*article_id*, *author_code*)
FK: article_id → Article(article_id) FK: author_code → Author(code)

ArticleAreaIndication(*article_id*, *area_acronym*)
FK: article_id → Article(article_id) FK: area_acronym → ResearchArea(area_acronym)

Partner(code, name, address)

Contribution(*article_id*, *partner_code*)
FK: article_id → Article(article_id) where category = 'Industrial Paper'
FK: partner_code → Partner(code)

Review(code, review_date, content, originality, significance, quality, global_score, comments, *article_id*, *reviewer_code*)
FK: article_id → Article(article_id) FK: reviewer_code → Organizer(code)

Info

- avg_global_score in Article is a derived redundant attribute, justified by the redundancy analysis in Section 3.3.
- The Reviewer specialization is resolved through the Review relationship: any organizer who has a review assigned is effectively a reviewer. No flag is needed.
- Article subtypes (Tutorial, Short Paper, etc.) are merged into a single Article table via the category attribute.
- The AreaIndication bridge keeps the original ternary decomposition through a three-component composite primary key.

Chapter 4

Physical Implementation

In this chapter we turn the logical schema into real Oracle SQL code. We write `CREATE TABLE` statements with constraints, triggers for the business rules, and stored procedures for the main operations.

4.1 Tables Definition

Tables are created in the right order: independent entities first, then the bridge tables that link them. All constraints (CHECK, NOT NULL, primary and foreign keys) are written inline.

Core Independent Entities

```
CREATE TABLE Sponsor ( -- just the name as PK
    name VARCHAR2(100) PRIMARY KEY
);

CREATE TABLE Conference ( -- main conference table
    acronym       VARCHAR2(50)  PRIMARY KEY,
    name          VARCHAR2(100) NOT NULL,
    location      VARCHAR2(100),
    homepage_url  VARCHAR2(150)
);

CREATE TABLE Organizer ( -- committee member / reviewer
    code         VARCHAR2(20)  PRIMARY KEY,
    name         VARCHAR2(100) NOT NULL,
    affiliation  VARCHAR2(100),
    address      VARCHAR2(200),
    phone        VARCHAR2(20),
    email        VARCHAR2(100)
);

CREATE TABLE ResearchArea ( -- research areas for matching
    area_acronym VARCHAR2(20)  PRIMARY KEY,
    area_name    VARCHAR2(100) NOT NULL,
    description  VARCHAR2(500)
);

```

```sql
CREATE TABLE Author ( -- paper authors
    code         VARCHAR2(20)  PRIMARY KEY,
    name         VARCHAR2(100) NOT NULL,
    affiliation  VARCHAR2(100),
    address      VARCHAR2(200),
    phone        VARCHAR2(20),
    email        VARCHAR2(100)
);

CREATE TABLE Partner ( -- industrial partners, BR12
    code     VARCHAR2(20)  PRIMARY KEY,
    name     VARCHAR2(100) NOT NULL,
    address  VARCHAR2(200)
);
```

Article Entity

```sql
CREATE TABLE Article (
    article_id           NUMBER GENERATED BY DEFAULT AS IDENTITY
       PRIMARY KEY,
    conference_acronym   VARCHAR2(50) NOT NULL,
    seq_number           INT          NOT NULL,
    title                VARCHAR2(200) NOT NULL,
    category             VARCHAR2(50)  NOT NULL
        CHECK (category IN ('Industrial Paper', 'Research Paper',
                            'Tutorial', 'Poster', 'Short Paper')), --
                               BR4
    status               VARCHAR2(20) DEFAULT 'pending'
        CHECK (status IN ('pending', 'accepted', 'rejected')), -- BR10
    avg_global_score     FLOAT DEFAULT 0.0, -- updated by trigger BR13
    contact_author_code  VARCHAR2(20) NOT NULL, -- must be in
       Authorship (BR8)
    FOREIGN KEY (conference_acronym) REFERENCES Conference(acronym),
    FOREIGN KEY (contact_author_code) REFERENCES Author(code),
    CONSTRAINT unique_conference_seq UNIQUE (conference_acronym,
       seq_number)
);
```

Bridge Relations (N:M Mappings)

```sql
CREATE TABLE Sponsorship ( -- sponsor <-> conference link
    sponsor_name        VARCHAR2(100),
    conference_acronym  VARCHAR2(50),
    funding_date        DATE   NOT NULL,
    funded_amount       NUMBER NOT NULL CHECK (funded_amount > 0), --
       BR7
    PRIMARY KEY (sponsor_name, conference_acronym),
    FOREIGN KEY (sponsor_name)       REFERENCES Sponsor(name),
```

```
    FOREIGN KEY (conference_acronym) REFERENCES Conference(acronym)
);

CREATE TABLE Membership ( -- committee membership, BR9 checks min 2
    organizer_code     VARCHAR2(20),
    conference_acronym VARCHAR2(50),
    PRIMARY KEY (organizer_code, conference_acronym),
    FOREIGN KEY (organizer_code)     REFERENCES Organizer(code),
    FOREIGN KEY (conference_acronym) REFERENCES Conference(acronym)
);

CREATE TABLE AreaIndication ( -- 3-way bridge: org+conf+area
    organizer_code     VARCHAR2(20),
    conference_acronym VARCHAR2(50),
    area_acronym       VARCHAR2(20),
    PRIMARY KEY (organizer_code, conference_acronym, area_acronym),
    FOREIGN KEY (organizer_code)     REFERENCES Organizer(code),
    FOREIGN KEY (conference_acronym) REFERENCES Conference(acronym),
    FOREIGN KEY (area_acronym)       REFERENCES ResearchArea(
        area_acronym)
);

CREATE TABLE Authorship ( -- who wrote which article
    article_id  NUMBER,
    author_code VARCHAR2(20),
    PRIMARY KEY (article_id, author_code),
    FOREIGN KEY (article_id)  REFERENCES Article(article_id) ON DELETE
        CASCADE,
    FOREIGN KEY (author_code) REFERENCES Author(code)
);

CREATE TABLE ArticleAreaIndication ( -- article <-> area link
    article_id   NUMBER,
    area_acronym VARCHAR2(20),
    PRIMARY KEY (article_id, area_acronym),
    FOREIGN KEY (article_id)   REFERENCES Article(article_id) ON
        DELETE CASCADE,
    FOREIGN KEY (area_acronym) REFERENCES ResearchArea(area_acronym)
);

CREATE TABLE Contribution ( -- partner contribution, only for
    Industrial
    article_id   NUMBER,
    partner_code VARCHAR2(20),
    PRIMARY KEY (article_id, partner_code),
    FOREIGN KEY (article_id)   REFERENCES Article(article_id) ON
        DELETE CASCADE,
    FOREIGN KEY (partner_code) REFERENCES Partner(code)
);
```

Review Entity

```
CREATE TABLE Review ( -- the actual review, scores 0-10
    code           VARCHAR2(20) PRIMARY KEY,
    review_date    DATE         NOT NULL,
    content        CLOB,
    originality    INT  CHECK (originality  BETWEEN 0 AND 10), -- BR10
       range
    significance   INT  CHECK (significance BETWEEN 0 AND 10),
    quality        INT  CHECK (quality      BETWEEN 0 AND 10),
    global_score   INT  CHECK (global_score BETWEEN 0 AND 10),
    comments       CLOB,
    article_id     NUMBER       NOT NULL, -- FK to Article
    reviewer_code VARCHAR2(20) NOT NULL, -- FK to Organizer (reviewer)
    FOREIGN KEY (article_id)    REFERENCES Article(article_id) ON
       DELETE CASCADE,
    FOREIGN KEY (reviewer_code) REFERENCES Organizer(code)
);
```

4.2 Triggers and Business Rules

We use Oracle PL/SQL triggers to enforce the business rules that cannot be done with simple CHECK constraints or foreign keys. Each trigger checks one specific rule before or after an operation on a table. The trigger names follow the pattern `trg_brN_description` so it is easy to see which rule each trigger enforces.

Info

To write the triggers, I studied the Oracle PL/SQL documentation (especially the sections on row-level triggers and compound triggers) to understand the mutating table problem and how compound triggers solve it.

Trigger: BR1 - Maximum Reviewer Cardinality

```
CREATE OR REPLACE TRIGGER trg_br1_max_reviewers
BEFORE INSERT ON Review  -- before we insert on review for each row
FOR EACH ROW
DECLARE
    v_count INT; -- aux variable we need to store the reviewer count
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM Review
    WHERE article_id = :NEW.article_id; --simple review count select

    IF v_count >= 4 THEN --throw error abort this operation
        RAISE_APPLICATION_ERROR(-20001,
          'BR1: An article cannot have more than 4 reviewers.');
    END IF;
END;
/
```

Trigger: BR2 - Reviewer Conference Membership

```
CREATE OR REPLACE TRIGGER trg_br2_reviewer_conf_match
BEFORE INSERT OR UPDATE ON Review  -- fires on insert or update
FOR EACH ROW
DECLARE
    v_article_conf         VARCHAR2(50); -- conference of the article
    v_reviewer_conf_count INT; -- check if reviewer is member
BEGIN
    SELECT conference_acronym INTO v_article_conf
    FROM Article WHERE article_id = :NEW.article_id; -- get the
        conference

    SELECT COUNT(*) INTO v_reviewer_conf_count
    FROM Membership
    WHERE organizer_code     = :NEW.reviewer_code
      AND conference_acronym = v_article_conf; -- is this reviewer in
          the committee?
```

```
15
16      IF v_reviewer_conf_count = 0 THEN -- not a member, block it
17          RAISE_APPLICATION_ERROR(-20002,
18            'BR2: Reviewer must be in the program committee.');
19      END IF;
20  END;
21  /
```

Trigger: BR3 - Research Area Compatibility

```
1   CREATE OR REPLACE TRIGGER trg_br3_reviewer_area_match
2   BEFORE INSERT OR UPDATE ON Review  -- check area compatibility
3   FOR EACH ROW
4   DECLARE
5       v_overlap INT; -- how many areas in common
6   BEGIN
7       SELECT COUNT(*) INTO v_overlap
8       FROM AreaIndication ai
9       JOIN ArticleAreaIndication aai
10        ON aai.area_acronym = ai.area_acronym
11      JOIN Article art
12        ON aai.article_id = art.article_id
13      WHERE ai.organizer_code = :NEW.reviewer_code
14        AND aai.article_id    = :NEW.article_id
15        AND ai.conference_acronym
16                              = art.conference_acronym; -- join to find
                                    overlap
17      IF v_overlap = 0 THEN -- no overlap means no match
18          RAISE_APPLICATION_ERROR(-20003,
19            'BR3: Reviewer areas must overlap with article areas.');
20      END IF;
21  END;
22  /
```

Info

BR4 (score range $[0, 10]$) and BR10 (positive sponsorship amount) are enforced declaratively via inline `CHECK` constraints on the respective table columns. BR7 (single conference submission) is enforced by the foreign key design of Article.

Trigger: BR5 & BR6 - Status Determination

```
1   CREATE OR REPLACE TRIGGER trg_br5_br6_article_status
2   BEFORE UPDATE OF status ON Article  -- only fires when status changes
3   FOR EACH ROW
4   DECLARE
5       v_total_reviews     INT; -- total reviews assigned
6       v_completed_reviews INT; -- reviews that have a global_score
```

```
BEGIN
    IF :OLD.status = 'pending'
       AND :NEW.status != 'pending'
    THEN -- only when leaving pending
        SELECT COUNT(*) INTO v_total_reviews
        FROM Review
        WHERE article_id = :NEW.article_id; -- count all reviews

        SELECT COUNT(*) INTO v_completed_reviews
        FROM Review
        WHERE article_id = :NEW.article_id
          AND global_score IS NOT NULL; -- count only completed ones

        IF v_total_reviews < 2 THEN -- need at least 2 reviews
            RAISE_APPLICATION_ERROR(-20006,
              'BR6: Min 2 reviews required.');
        END IF;
        IF v_total_reviews != v_completed_reviews THEN -- all must be
            done
            RAISE_APPLICATION_ERROR(-20005,
              'BR5: All reviews must be done.');
        END IF;
    END IF;
END;
/
```

Trigger: BR8 - Contact Author Subset

```
CREATE OR REPLACE TRIGGER trg_br8_contact_is_author
AFTER INSERT OR UPDATE ON Article  -- after because authorship comes
    later
FOR EACH ROW
DECLARE
    v_is_author INT; -- check if contact is in authorship table
BEGIN
    SELECT COUNT(*) INTO v_is_author
    FROM Authorship
    WHERE article_id  = :NEW.article_id
      AND author_code = :NEW.contact_author_code; -- is this person an
          author?
    IF v_is_author = 0 AND NOT INSERTING THEN -- skip on insert,
        authorship comes after
        RAISE_APPLICATION_ERROR(-20008,
          'BR8: Contact author must be one of the authors.');
    END IF;
END;
/
```

Trigger: BR11 - Acceptance Score Threshold

```
CREATE OR REPLACE TRIGGER trg_br11_acceptance_threshold
BEFORE UPDATE OF status ON Article  -- check before accepting
FOR EACH ROW
BEGIN
    IF :NEW.status = 'accepted'
       AND :NEW.avg_global_score < 5
    THEN -- score too low
        RAISE_APPLICATION_ERROR(-20011,
          'BR11: avg score must be >= 5 to accept.');
    END IF;
END;
/
```

Trigger: BR12 - Industrial Paper Only

```
CREATE OR REPLACE TRIGGER trg_br12_industrial_paper_only
BEFORE INSERT OR UPDATE ON Contribution  -- check before linking
   partner
FOR EACH ROW
DECLARE
    v_category VARCHAR2(50); -- article category
BEGIN
    SELECT category INTO v_category
    FROM Article
    WHERE article_id = :NEW.article_id; -- what kind of article is
       this?
    IF v_category != 'Industrial Paper' THEN -- not industrial? block
        RAISE_APPLICATION_ERROR(-20012,
          'BR12: Partners can only contribute to Industrial Papers.');
    END IF;
END;
/
```

4.2.1 Compound Triggers (BR9 and BR13)

The two triggers below use the Oracle compound trigger syntax. This is needed when a trigger on a table also needs to query or update the same table. In a normal row-level trigger, Oracle raises the "mutating table" error (ORA-04091) if you try to read the table being modified. The compound trigger solves this by splitting the logic into two phases:

1. **AFTER EACH ROW**: collects the keys of the affected rows into a temporary array.
2. **AFTER STATEMENT**: after all rows have been processed, reads the table (which is now stable) and applies the validation logic.

Trigger: BR9 - Minimum Committee Size (Compound)

```
CREATE OR REPLACE TRIGGER trg_br9_min_committee
FOR DELETE ON Membership  -- fires when we remove someone from
    committee
COMPOUND TRIGGER
    TYPE t_confs IS TABLE OF VARCHAR2(50)
      INDEX BY PLS_INTEGER;
    g_confs t_confs; -- collect affected conferences
    g_idx   PLS_INTEGER := 0;

    AFTER EACH ROW IS
    BEGIN
        g_idx := g_idx + 1;
        g_confs(g_idx) := :OLD.conference_acronym; -- save which
            conference
    END AFTER EACH ROW;

    AFTER STATEMENT IS
        v_count INT; -- remaining members count
    BEGIN
        FOR i IN 1..g_idx LOOP
            SELECT COUNT(*) INTO v_count
            FROM Membership
            WHERE conference_acronym = g_confs(i); -- how many left?
            IF v_count < 2 THEN -- too few, block
                RAISE_APPLICATION_ERROR(-20009,
                  'BR9: Min 2 committee members required.');
            END IF;
        END LOOP;
    END AFTER STATEMENT;
END trg_br9_min_committee;
/
```

Trigger: BR13 - Avg Score Maintenance (Compound)

```
CREATE OR REPLACE TRIGGER trg_br13_update_avg_score
FOR INSERT OR UPDATE OF global_score
    OR DELETE ON Review  -- fires on any score change
COMPOUND TRIGGER
    TYPE t_ids IS TABLE OF NUMBER
      INDEX BY PLS_INTEGER;
    g_ids t_ids; -- collect affected article ids
    g_idx PLS_INTEGER := 0;

    AFTER EACH ROW IS
    BEGIN
        g_idx := g_idx + 1;
        IF DELETING THEN
            g_ids(g_idx) := :OLD.article_id; -- deleted row, use OLD
```

```
        ELSE
            g_ids(g_idx) := :NEW.article_id; -- inserted/updated, use
                NEW
        END IF;
    END AFTER EACH ROW;

    AFTER STATEMENT IS
        v_avg FLOAT; -- new average
    BEGIN
        FOR i IN 1..g_idx LOOP
            SELECT NVL(AVG(global_score), 0)
            INTO v_avg
            FROM Review
            WHERE article_id = g_ids(i); -- recalculate avg

            UPDATE Article
            SET avg_global_score = v_avg
            WHERE article_id = g_ids(i); -- save it back
        END LOOP;
    END AFTER STATEMENT;
END trg_br13_update_avg_score;
/
```

4.3 Procedures

These PL/SQL stored procedures do the main operations (OP1–OP4). Each procedure handles its own transaction with COMMIT and ROLLBACK.

Procedure: Submit a New Article (OP1)

```
CREATE OR REPLACE PROCEDURE prc_submit_article (
    p_conf_acronym   IN  VARCHAR2, -- which conference
    p_title          IN  VARCHAR2,
    p_category       IN  VARCHAR2,
    p_contact_code   IN  VARCHAR2, -- contact author code
    p_out_article_id OUT NUMBER    -- returns the new article id
)
AS
    v_seq NUMBER; -- next seq number for this conference
BEGIN
    SELECT NVL(MAX(seq_number), 0) + 1 INTO v_seq
    FROM Article
    WHERE conference_acronym = p_conf_acronym; -- get next seq

    INSERT INTO Article (
        conference_acronym, seq_number,
        title, category, status,
        contact_author_code
    ) VALUES (
        p_conf_acronym, v_seq,
        p_title, p_category, 'pending',
        p_contact_code
    ) RETURNING article_id INTO p_out_article_id; -- grab the
       generated id

    INSERT INTO Authorship (article_id, author_code)
    VALUES (p_out_article_id, p_contact_code); -- also add as author

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

Procedure: Assign a Reviewer to an Article

```
CREATE OR REPLACE PROCEDURE prc_assign_reviewer (
    p_article_id     IN NUMBER,
    p_reviewer_code  IN VARCHAR2
)
AS
    v_code VARCHAR2(20); -- auto-generated review code
```

```
BEGIN
    v_code := 'REV-' || p_article_id
              || '-' || SUBSTR(p_reviewer_code, 1, 4); -- build the
                  code

    INSERT INTO Review (
        code, review_date, article_id, reviewer_code
    ) VALUES (
        v_code, SYSDATE, p_article_id, p_reviewer_code
    ); -- triggers BR1, BR2, BR3 will fire here

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

Procedure: Submit a Review (OP3)

```
CREATE OR REPLACE PROCEDURE prc_add_review (
    p_code         IN VARCHAR2, -- review code (already created by
        assign)
    p_originality  IN INT,
    p_significance IN INT,
    p_quality      IN INT,
    p_comments     IN CLOB,
    p_content      IN CLOB
)
AS
    v_global_score INT; -- calculated average of the 3 scores
BEGIN
    v_global_score := ROUND(
        (p_originality + p_significance + p_quality) / 3
    ); -- simple avg, rounded

    UPDATE Review
    SET review_date  = SYSDATE,
        content      = p_content,
        originality  = p_originality,
        significance = p_significance,
        quality      = p_quality,
        global_score = v_global_score, -- this triggers BR13 to update
            avg
        comments     = p_comments
    WHERE code = p_code;

    COMMIT;
```

```
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

Procedure: Get Reviewer Assignments (OP2)

```
CREATE OR REPLACE PROCEDURE prc_get_reviewer_assignments (
    p_reviewer_code IN  VARCHAR2,
    p_cursor        OUT SYS_REFCURSOR -- returns the result set
)
AS
BEGIN
    OPEN p_cursor FOR
        SELECT
            art.title,
            art.category,
            ( -- subquery to get all authors as comma-separated list
                SELECT LISTAGG(au.name, ', ')
                  WITHIN GROUP (ORDER BY au.name)
                FROM Authorship auth
                JOIN Author au ON auth.author_code = au.code
                WHERE auth.article_id = art.article_id
            ) AS authors_list
        FROM Review rev
        JOIN Article art ON rev.article_id = art.article_id
        WHERE rev.reviewer_code = p_reviewer_code; -- filter by
           reviewer
END;
/
```

Procedure: Get Accepted Articles per Conference (OP4)

```
CREATE OR REPLACE PROCEDURE prc_get_accepted_articles (
    p_acronym IN  VARCHAR2,
    p_cursor  OUT SYS_REFCURSOR -- returns accepted articles
)
AS
BEGIN
    OPEN p_cursor FOR
        SELECT
            art.category,
            art.title,
            a.name  AS contact_author_name,
            a.email AS contact_author_email,
            art.avg_global_score
```

```
        FROM Article art
        JOIN Author a ON art.contact_author_code = a.code
        WHERE art.conference_acronym = p_acronym
          AND art.status = 'accepted' -- only accepted ones
        ORDER BY art.category, art.title;
END;
/
```

4.4 Utility Constructors

These two utility procedures help set up entities that need multiple tables at once. They are used by the population script and the web app.

Utility: Initialize a Conference with Committee

```
CREATE OR REPLACE PROCEDURE util_init_conference (
    p_acronym     IN VARCHAR2,
    p_name        IN VARCHAR2,
    p_location    IN VARCHAR2,
    p_url         IN VARCHAR2,
    p_org_codes   IN VARCHAR2,   -- comma-separated organizer codes
    p_area_acr    IN VARCHAR2    -- comma-separated area acronyms
)
AS
    v_org    VARCHAR2(20);   -- current organizer being parsed
    v_area   VARCHAR2(20);   -- current area being parsed
    v_pos    INT;            -- comma position
    v_str    VARCHAR2(4000); -- working string for parsing
BEGIN
    -- create the conference first
    INSERT INTO Conference (acronym, name, location, homepage_url)
    VALUES (p_acronym, p_name, p_location, p_url);

    -- parse organizer codes one by one
    v_str := p_org_codes || ',';
    WHILE INSTR(v_str, ',') > 0 LOOP
        v_pos := INSTR(v_str, ',');
        v_org := TRIM(SUBSTR(v_str, 1, v_pos - 1));
        v_str := SUBSTR(v_str, v_pos + 1);

        INSERT INTO Membership (organizer_code, conference_acronym)
        VALUES (v_org, p_acronym); -- add to committee

        -- now assign all areas to this organizer
        DECLARE
            v_astr VARCHAR2(4000) := p_area_acr || ',';
            v_apos INT;
        BEGIN
            WHILE INSTR(v_astr, ',') > 0 LOOP
                v_apos := INSTR(v_astr, ',');
                v_area := TRIM(SUBSTR(v_astr, 1, v_apos - 1));
                v_astr := SUBSTR(v_astr, v_apos + 1);
                BEGIN
                    INSERT INTO AreaIndication
                        (organizer_code, conference_acronym,
                           area_acronym)
                    VALUES (v_org, p_acronym, v_area);
                EXCEPTION WHEN DUP_VAL_ON_INDEX THEN NULL; -- skip
                   dups
                END;
```

```
            END LOOP;
        END;
    END LOOP;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

Utility: Register an Article with Authors and Areas

```
CREATE OR REPLACE PROCEDURE util_register_article (
    p_conf_acronym    IN  VARCHAR2,
    p_title           IN  VARCHAR2,
    p_category        IN  VARCHAR2,
    p_contact_code    IN  VARCHAR2,
    p_author_codes    IN  VARCHAR2,   -- comma-separated author codes
    p_area_acronyms   IN  VARCHAR2,   -- comma-separated area acronyms
    p_out_article_id OUT NUMBER       -- returns the new article id
)
AS
    v_seq  NUMBER;          -- next seq number
    v_code VARCHAR2(20);    -- current author code being parsed
    v_area VARCHAR2(20);    -- current area being parsed
    v_pos  INT;             -- comma position
    v_str  VARCHAR2(4000); -- working string
BEGIN
    -- get next seq number for this conference
    SELECT NVL(MAX(seq_number), 0) + 1 INTO v_seq
    FROM Article WHERE conference_acronym = p_conf_acronym;

    -- insert the article
    INSERT INTO Article (
        conference_acronym, seq_number, title, category,
        status, contact_author_code
    ) VALUES (
        p_conf_acronym, v_seq, p_title, p_category,
        'pending', p_contact_code
    ) RETURNING article_id INTO p_out_article_id; -- grab the id

    -- parse and add each author
    v_str := p_author_codes || ',';
    WHILE INSTR(v_str, ',') > 0 LOOP
        v_pos := INSTR(v_str, ',');
        v_code := TRIM(SUBSTR(v_str, 1, v_pos - 1));
        v_str := SUBSTR(v_str, v_pos + 1);
```

```
        BEGIN
            INSERT INTO Authorship (article_id, author_code)
            VALUES (p_out_article_id, v_code);
        EXCEPTION WHEN DUP_VAL_ON_INDEX THEN NULL; -- skip dups
        END;
    END LOOP;

    -- parse and add each area
    v_str := p_area_acronyms || ',';
    WHILE INSTR(v_str, ',') > 0 LOOP
        v_pos := INSTR(v_str, ',');
        v_area := TRIM(SUBSTR(v_str, 1, v_pos - 1));
        v_str := SUBSTR(v_str, v_pos + 1);
        BEGIN
            INSERT INTO ArticleAreaIndication (article_id,
               area_acronym)
            VALUES (p_out_article_id, v_area);
        EXCEPTION WHEN DUP_VAL_ON_INDEX THEN NULL; -- skip dups
        END;
    END LOOP;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

4.5 Database Population Data

This PL/SQL script fills the database with test data that matches the volume table from Chapter 3. It uses round-robin patterns to spread the data evenly.

Info

The script uses Oracle `IDENTITY` for auto-generated IDs, `RETURNING INTO` to get the generated keys, and `BEGIN...EXCEPTION` blocks to skip duplicates.

PL/SQL System Volume Populator Script (Condensed)

```
SET SERVEROUTPUT ON;
DECLARE
    v_conf_acronym VARCHAR2(50);
    v_org_code     VARCHAR2(20);
    v_author_code  VARCHAR2(20);
    v_article_id   NUMBER;
    v_review_code  VARCHAR2(20);
    v_category     VARCHAR2(50);
    v_area_acronym VARCHAR2(20);
BEGIN
    -- 1. Seed research areas (3 areas)
    INSERT INTO ResearchArea (area_acronym, area_name, description)
    VALUES ('DB_SYS', 'Database Systems', 'Storage and querying.');
    INSERT INTO ResearchArea (area_acronym, area_name, description)
    VALUES ('SW_ENG', 'Software Engineering', 'Testing and
       architecture.');
    INSERT INTO ResearchArea (area_acronym, area_name, description)
    VALUES ('AI', 'Artificial Intelligence', 'ML, reasoning, agents.')
       ;

    -- 2. Sponsors (20), Partners (480), Authors (3600)
    FOR i IN 1..20 LOOP
        INSERT INTO Sponsor (name) VALUES ('Sponsor_' || i);
    END LOOP;
    FOR i IN 1..480 LOOP
        INSERT INTO Partner (code, name, address)
        VALUES ('P_' || i, 'Partner Corp ' || i, 'City ' || i);
    END LOOP;
    FOR i IN 1..3600 LOOP
        INSERT INTO Author (
            code, name, affiliation,
            address, phone, email
        ) VALUES (
            'A_' || i, 'Author ' || i,
            'Uni_' || MOD(i, 50),
            'Addr ' || i, '555-' || i,
            'auth' || i || '@test.com'
        );
    END LOOP;
```

```
38
39      -- 3. Main Conference Loop (100 Conferences)
40      FOR c IN 1..100 LOOP
41          v_conf_acronym := 'CONF_' || c;
42          INSERT INTO Conference (acronym, name, location, homepage_url)
43          VALUES (v_conf_acronym, 'Conference ' || c,
44                  'City_' || c, 'http://conf' || c || '.org');
45
46          -- 5 sponsors per conference
47          FOR s IN 1..5 LOOP
48              INSERT INTO Sponsorship (sponsor_name, conference_acronym,
49                                       funding_date, funded_amount)
50              VALUES ('Sponsor_' || (MOD((c-1)*5+s-1, 20)+1),
51                      v_conf_acronym, SYSDATE, 10000 + s * 1000);
52          END LOOP;
53
54          -- 4 organizers per conference (400 total)
55          FOR o IN 1..4 LOOP
56              v_org_code := 'ORG_' || c || '_' || o;
57              INSERT INTO Organizer (
58                  code, name, affiliation,
59                  address, phone, email
60              ) VALUES (
61                  v_org_code,
62                  'Org ' || v_org_code,
63                  'Affil_' || MOD(o, 10),
64                  'Addr_' || o,
65                  '123-' || o,
66                  'org' || v_org_code || '@test.com'
67              );
68              INSERT INTO Membership (organizer_code, conference_acronym
                    )
69              VALUES (v_org_code, v_conf_acronym);
70              v_area_acronym := CASE MOD(o, 3) WHEN 0 THEN 'DB_SYS'
71                  WHEN 1 THEN 'SW_ENG' ELSE 'AI' END;
72              INSERT INTO AreaIndication (organizer_code,
73                  conference_acronym, area_acronym)
74              VALUES (v_org_code, v_conf_acronym, v_area_acronym);
75          END LOOP;
76
77          -- 12 articles per conference (1200 total)
78          FOR a IN 1..12 LOOP
79              IF a <= 2 THEN v_category := 'Industrial Paper';
80              ELSIF a <= 4 THEN v_category := 'Tutorial';
81              ELSIF a <= 5 THEN v_category := 'Short Paper';
82              ELSIF a <= 6 THEN v_category := 'Poster';
83              ELSE v_category := 'Research Paper'; END IF;
84
85              v_author_code := 'A_' || ((c - 1) * 12 + a);
86              v_area_acronym := CASE MOD(a, 3)
```

```
            WHEN 0 THEN 'DB_SYS'
            WHEN 1 THEN 'SW_ENG' ELSE 'AI' END;

        INSERT INTO Article (conference_acronym, seq_number,
            title, category, status, contact_author_code)
        VALUES (v_conf_acronym, a,
            'Article ' || c || '_' || a,
            v_category, 'pending', v_author_code)
        RETURNING article_id INTO v_article_id;

        -- 3 authors per article
        INSERT INTO Authorship (article_id, author_code)
        VALUES (v_article_id, v_author_code);
        INSERT INTO Authorship (article_id, author_code)
        VALUES (v_article_id,
            'A_' || (MOD((c-1)*12+a, 3600)+1));
        INSERT INTO Authorship (article_id, author_code)
        VALUES (v_article_id,
            'A_' || (MOD((c-1)*12+a+1, 3600)+1));

        -- Article area indication (2 areas per article)
        INSERT INTO ArticleAreaIndication (article_id,
           area_acronym)
        VALUES (v_article_id, v_area_acronym);
        BEGIN
            INSERT INTO ArticleAreaIndication (article_id,
               area_acronym)
            VALUES (v_article_id, CASE v_area_acronym
                WHEN 'DB_SYS' THEN 'SW_ENG'
                WHEN 'SW_ENG' THEN 'AI'
                ELSE 'DB_SYS' END);
        EXCEPTION WHEN DUP_VAL_ON_INDEX THEN NULL;
        END;

        -- Contribution for Industrial Papers only
        IF v_category = 'Industrial Paper' THEN
            INSERT INTO Contribution (article_id, partner_code)
            VALUES (v_article_id, 'P_' || (MOD(a,480)+1));
            INSERT INTO Contribution (article_id, partner_code)
            VALUES (v_article_id, 'P_' || (MOD(a+1,480)+1));
        END IF;

        -- 3 reviews per article
        FOR r IN 1..3 LOOP
            v_review_code := 'REV_' || v_article_id || '_' || r;
            v_org_code := 'ORG_' || c || '_' || r;
            BEGIN
                INSERT INTO AreaIndication (organizer_code,
                    conference_acronym, area_acronym)
```

```
                    VALUES (v_org_code, v_conf_acronym, v_area_acronym
                        );
                EXCEPTION WHEN DUP_VAL_ON_INDEX THEN NULL;
                END;
                INSERT INTO Review (
                    code, review_date, content,
                    originality, significance,
                    quality, global_score,
                    article_id, reviewer_code
                ) VALUES (
                    v_review_code, SYSDATE,
                    'Review ' || v_article_id,
                    5+MOD(r,5),
                    5+MOD(r+1,5),
                    5+MOD(r+2,5),
                    ROUND((5+MOD(r,5)
                      +5+MOD(r+1,5)
                      +5+MOD(r+2,5))/3),
                    v_article_id, v_org_code
                );
            END LOOP;
        END LOOP;
    END LOOP;
    COMMIT;
END;
/
```

4.6 Trigger Testing

To check that all triggers work as expected, we wrote a testing script that tries both valid and invalid operations for each business rule. Each test is wrapped in a `BEGIN...EXCEPTION` block so it does not stop the script when an error is raised. The test checks `SQLCODE` to confirm it matches the expected error number.

Info

The full testing script is in `sql/05_testing.sql`. It should be run after the tables, triggers, and population script. Individual trigger definitions are also available as standalone files inside `sql/triggers/`.

Here is an example of how we test BR1 (max 4 reviewers):

Example: Testing BR1

```
-- Article 1 already has 3 reviews from the population script.
-- A 4th review should be accepted, a 5th should be blocked.
BEGIN
    INSERT INTO Review (code, review_date, article_id, reviewer_code)
    VALUES ('TEST_BR1_4', SYSDATE, 1, 'ORG_1_4');
    -- This succeeds (4th reviewer)

    INSERT INTO Review (code, review_date, article_id, reviewer_code)
    VALUES ('TEST_BR1_5', SYSDATE, 1, 'ORG_TEST');
    -- This should raise ORA-20001
    ROLLBACK;
EXCEPTION
    WHEN OTHERS THEN
        IF SQLCODE = -20001 THEN
            DBMS_OUTPUT.PUT_LINE('BR1 PASS');
        END IF;
        ROLLBACK;
END;
/
```

The testing script covers all 13 business rules, including the compound triggers (BR9, BR13) and the CHECK constraints (BR4, BR10).

4.7 Index Analysis

Oracle automatically creates indexes for all PRIMARY KEY and UNIQUE columns. But it does not create indexes for FOREIGN KEY columns. Since many of our triggers and procedures filter on FK columns, we need to check the execution plans and add indexes where needed.

4.7.1 Methodology

For each critical query we follow these steps:

1. Identify the query and which triggers/procedures use it.
2. Run `EXPLAIN PLAN` to see how Oracle executes it without a custom index.
3. If the plan shows a `TABLE ACCESS FULL` (full scan), create an index.
4. Run `EXPLAIN PLAN` again to confirm the index is used.

Info

The full analysis script is in `sql/06_indexes.sql`. It uses `DBMS_XPLAN.DISPLAY` to show the execution plans before and after creating the indexes.

4.7.2 Query Q1: Review by article_id

This query appears in four triggers: BR1 (max reviewers), BR5/BR6 (status check), and BR13 (average score). It runs every time a review is inserted, updated, or deleted.

Q1: COUNT reviews for a given article

```
SELECT COUNT(*) FROM Review
WHERE article_id = :NEW.article_id;
```

The column `Review.article_id` is a foreign key but has no index. Oracle must scan all rows in Review to count how many belong to the article.

Assumption / Design Reasoning

`Review.article_id` is the most used FK column in the whole schema. It shows up in 4 triggers that fire on every DML on the Review table. With 3 600 review rows, a full table scan is not the end of the world, but it is not needed either. This is the top-priority index.

Plan BEFORE index:

```
---------------------------------------------
| Operation              | Name             |
---------------------------------------------
| SELECT STATEMENT       |                  |
|  SORT AGGREGATE        |                  |
|   TABLE ACCESS FULL    | REVIEW           |
---------------------------------------------
```

Plan AFTER index:

```
---------------------------------------------
| Operation                | Name           |
---------------------------------------------
| SELECT STATEMENT         |                |
|  SORT AGGREGATE          |                |
|   INDEX RANGE SCAN       | IDX_REVIEW_ART_ID |
---------------------------------------------
```

The full table scan is replaced by an `INDEX RANGE SCAN`, which reads only the index entries for the specific `article_id` value.

4.7.3 Query Q2: Review by reviewer_code

This query is used by OP2 (`prc_get_reviewer_assignments`) to find which articles are assigned to a reviewer.

Q2: Find reviews for a given reviewer

```sql
SELECT rev.article_id
FROM Review rev
WHERE rev.reviewer_code = 'ORG_1_1';
```

Assumption / Design Reasoning

`Review.reviewer_code` is a FK column with no index. Every time a reviewer opens the assignments page, Oracle scans the whole Review table. With an index, Oracle goes straight to the rows for that reviewer.

Plan BEFORE index:

```
---------------------------------------------
| Operation                | Name           |
---------------------------------------------
| SELECT STATEMENT         |                |
|  TABLE ACCESS FULL       | REVIEW         |
---------------------------------------------
```

Plan AFTER index:

```
-------------------------------------------------
| Operation                  | Name             |
-------------------------------------------------
| SELECT STATEMENT           |                  |
|  TABLE ACCESS BY ROWID     | REVIEW           |
|   INDEX RANGE SCAN         | IDX_REVIEW_REV_CD |
-------------------------------------------------
```

4.7.4 Query Q3: Membership by conference_acronym

This query is used by the BR9 compound trigger to check that a conference still has at least 2 committee members after a deletion.

Q3: Count members for a conference

```
SELECT COUNT(*) FROM Membership
WHERE conference_acronym = 'CONF_1';
```

Assumption / Design Reasoning

Membership's PK is `(organizer_code, conference_acronym)`. Since `conference_acronym` is the *second* column in the composite PK, Oracle cannot use the PK index to search by `conference_acronym` alone. This is how B-tree indexes work: they only support prefix-based lookups. We need a separate index with `conference_acronym` as the first column.

Plan BEFORE index:

```
-----------------------------------------------
| Operation                | Name               |
-----------------------------------------------
| SELECT STATEMENT         |                    |
|  SORT AGGREGATE          |                    |
|   TABLE ACCESS FULL      | MEMBERSHIP         |
-----------------------------------------------
```

Plan AFTER index:

```
-------------------------------------------------
| Operation                  | Name             |
-------------------------------------------------
| SELECT STATEMENT           |                  |
|  SORT AGGREGATE            |                  |
|   INDEX RANGE SCAN         | IDX_MEMB_CONF    |
-------------------------------------------------
```

4.7.5 Index Definitions

Based on the analysis above, we create three indexes:

Index Definitions

```
-- most used FK, fires in 4 triggers
CREATE INDEX idx_review_article_id
    ON Review(article_id);

-- for the assignments page (OP2)
CREATE INDEX idx_review_reviewer_code
    ON Review(reviewer_code);

-- conference_acronym is 2nd in PK, need separate index (BR9)
CREATE INDEX idx_membership_conf
    ON Membership(conference_acronym);
```

4.7.6 Summary

Table 4.1 shows all three indexes with the queries they speed up and the plan change.

Table 4.1: Custom indexes created for ConferenceHub.

Index	Table(Column)	Used by	Plan change
`idx_review_article_id`	Review(article_id)	BR1, BR5/6, BR13	FULL → RANGE SCAN
`idx_review_reviewer_code`	Review(reviewer_code)	OP2	FULL → RANGE SCAN
`idx_membership_conf`	Membership(conference_acronym)	BR9	FULL → RANGE SCAN

Assumption / Design Reasoning

We decided not to create indexes on columns that are already covered by existing PK or UNIQUE indexes. For example, `Authorship(article_id)` is the first column of the PK `(article_id, author_code)`, so Oracle can already use the PK index for queries that filter by `article_id`. Same thing for `AreaIndication` and `ArticleAreaIndication`.

Assumption / Design Reasoning

We also decided not to add too many indexes. Each extra index costs storage and slows down INSERT/UPDATE/DELETE because Oracle has to update both the table and the index. Our three indexes target the most important queries: the ones that fire on every Review operation (indexes 1 and 2) and the BR9 trigger (index 3).

Chapter 5

Web Application Architecture

This chapter describes the web application built on top of the Oracle database. The application is a simple Flask web server that calls stored procedures and displays results in HTML pages. All business logic stays in the database.

5.1 Architecture Overview

The application uses a three-tier architecture:

- **Presentation Tier**: HTML pages rendered with Jinja2 templates, styled with Tailwind CSS.
- **Application Tier**: Flask web server that routes HTTP requests to Oracle procedures.
- **Data Tier**: Oracle 21c database with tables, triggers, and stored procedures.

5.2 Flask Implementation

5.2.1 Database Connection

Flask connects to Oracle using the `python-oracledb` library in thin mode (no Oracle Client needed). Each HTTP request gets its own connection stored in Flask's `g` object:

```
def get_db():
    if 'db' not in g:
        g.db = oracledb.connect(
            user=DB_USER,
            password=DB_PASS,
            dsn=DB_DSN
        )
    return g.db

@app.teardown_appcontext
def close_connection(exception):
    db = g.pop('db', None)
    if db is not None:
        db.close()
```

5.2.2 Routes and SQL Procedures

The application has five routes that map to database operations:

Table 5.1: Flask routes and database operations.

URL	Procedure	Description
/	—	Home page
/conferences	direct SQL	List all conferences
/conference/<acr>/...	prc_get_accepted_articles	OP4: Accepted articles
/submit-review	prc_add_review	OP3: Submit a review
/reviewer/<code>/...	prc_get_reviewer_assignments	OP2: Reviewer assignments

Flask calls stored procedures using `cursor.callproc()` and retrieves results via `SYS_REFCURSOR`. For example, OP4 (accepted articles) works as follows:

```
ref = cur.var(oracledb.CURSOR)
cur.callproc('prc_get_accepted_articles', [acronym, ref])
articles = ref.getvalue().fetchall()
```

The procedure returns a cursor that Flask converts to a list of rows, which are then passed to the Jinja2 template for rendering.

5.3 Docker Deployment

The entire stack runs in Docker with two containers orchestrated by Docker Compose:

- **oracle-xe**: Oracle Database 21c Express Edition. Data persists in a named volume (`oracle-data`).
- **web**: Flask application built from `app/Dockerfile`. The `sql/` folder is mounted read-only.

5.3.1 Startup Process

The `entrypoint.sh` script in the web container:

1. Waits for Oracle to be ready (retry loop with 5-second intervals)
2. Executes `00_full_setup.sql` to create tables, triggers, procedures, and test data
3. Starts Gunicorn WSGI server with 2 worker processes on port 5000

5.3.2 Running the Application

From the project root:

```
docker compose up -build
```

The application is available at `http://localhost:5000`. First startup takes 3–5 minutes while Oracle initializes.

5.4 Screenshots

Below are some screenshots of the running application.

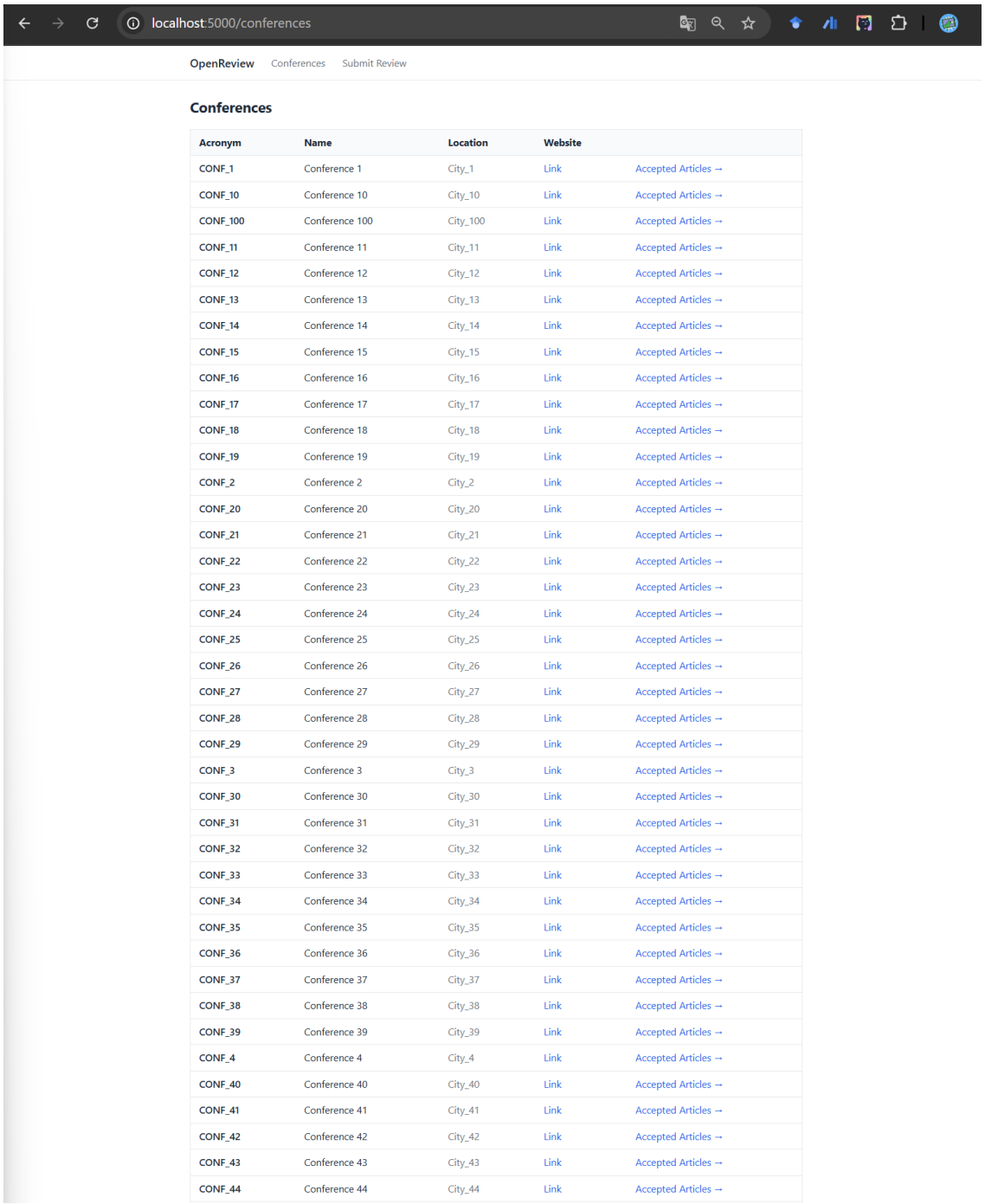

Figure 5.1: Conferences listing page.

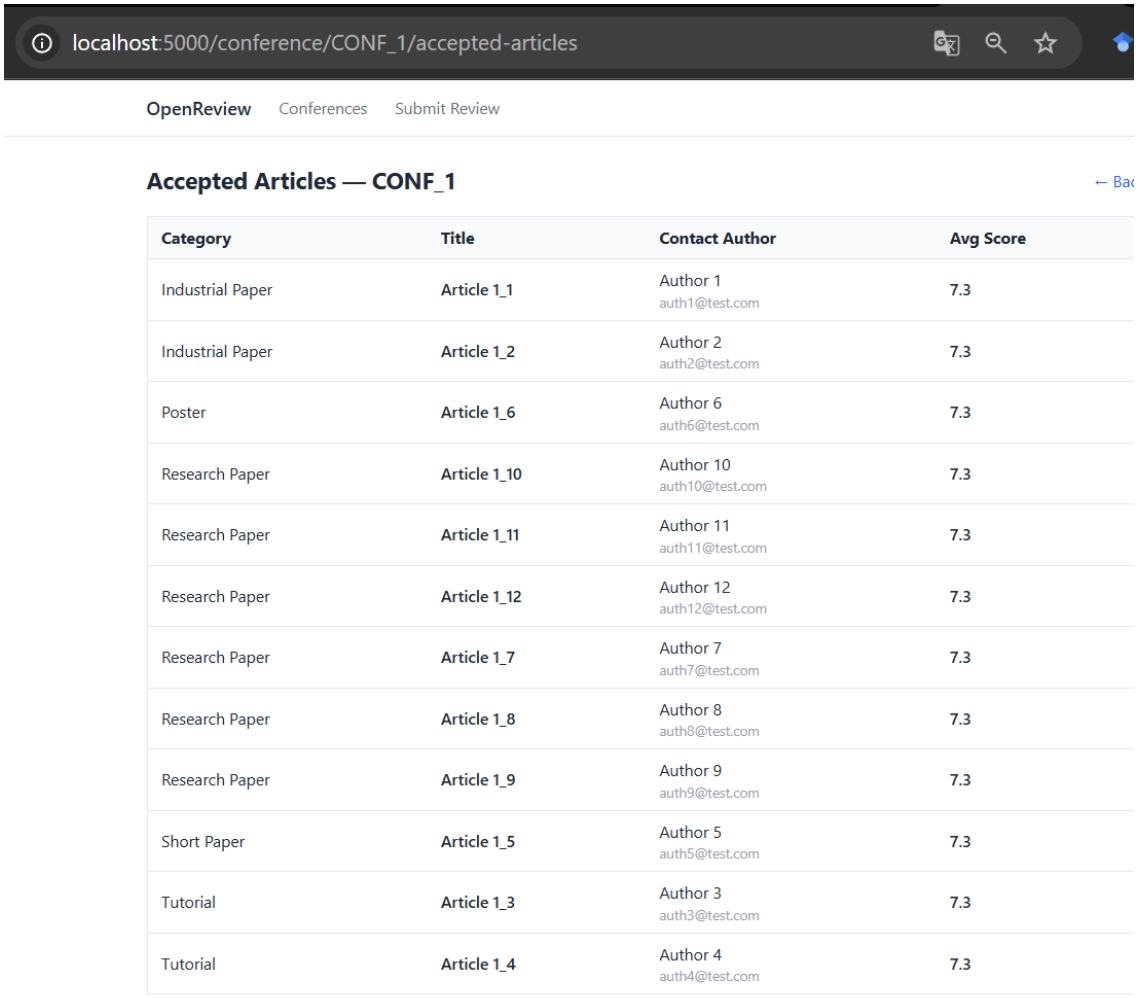

Figure 5.2: Accepted articles for a conference.

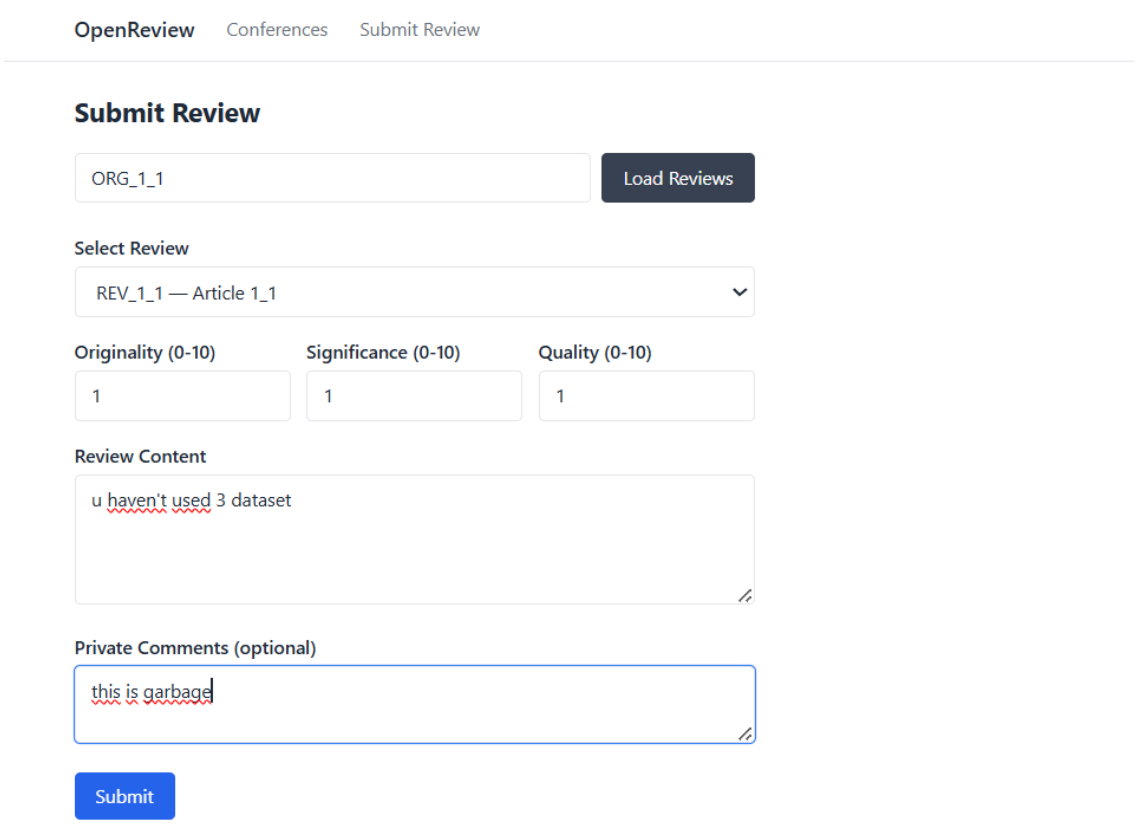

Figure 5.3: Submit review form.

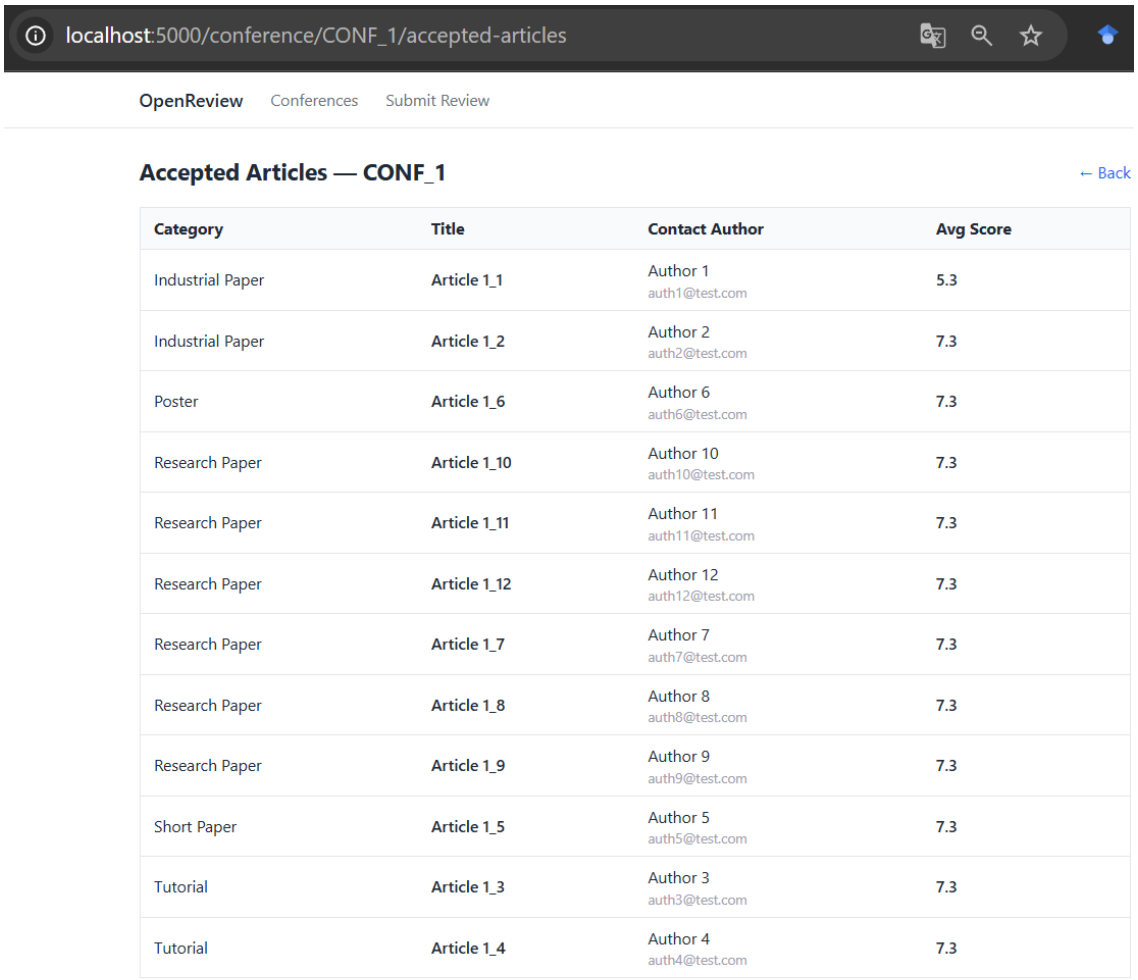

Figure 5.4: Accepted articles after reviews are submitted.

Chapter 6

Bonus: Data Warehouse

In this chapter we add a simple Data Warehouse (DW) layer on top of the ConferenceHub database. The idea is to have a separate, read-optimized structure that supports analytical queries without slowing down the main system.

6.1 Why a Data Warehouse

The database we built in the previous chapters follows the OLTP model: normalized tables, fast writes, strong constraints. This is good for daily transactions, but it is not great for analytical questions like "what is the acceptance rate per area and year?". These questions need many joins and aggregations that are expensive on a normalized schema.

A Data Warehouse solves this problem by storing data in a denormalized format, optimized for reading and aggregating. Table 6.1 shows the main differences.

Table 6.1: OLTP vs. OLAP comparison.

Property	OLTP (Our DB)	OLAP (Data Warehouse)
Purpose	Daily transactions	Analysis and reporting
Data model	Normalized (3NF)	Denormalized (Star Schema)
Queries	Short, targeted	Long, with aggregations
Updates	Frequent (INSERT/UPDATE)	Batch load (ETL)
Users	Application	Analysts, management

6.2 Star Schema Design

We use a Star Schema: one central Fact table surrounded by Dimension tables. This is the simplest and most common approach for data warehouses.

Info

In a star schema, dimension tables are fully denormalized. This means more storage space, but queries are simpler because each analytical query needs at most one join per dimension.

6.2.1 Grain

The grain defines what one row in the fact table represents. We chose:

- **FACT_REVIEW**: one row per review. This allows us to analyze scores at the individual review level.
- **FACT_SUBMISSION**: one row per article submission. This allows us to compute acceptance rates and submission volumes.

6.2.2 Dimensions

The dimension tables provide the axes along which we can slice and filter the data.

Table 6.3: Dimension tables.

Dimension	What it represents	Main Attributes
DIM_TIME	Calendar dates for time-based analysis	year, quarter, month
DIM_CONFERENCE	Conference details	acronym, name, location
DIM_RESEARCH_AREA	Research domains	acronym, description
DIM_CATEGORY	Article category	category_name
DIM_AFFILIATION	Reviewer/author institution	affiliation_name

6.2.3 Fact Tables

The fact tables store the measures (numbers we want to aggregate) and the foreign keys to the dimensions.

- **FACT_REVIEW**: stores individual review scores (originality, significance, quality, global). All scores are fully additive — we can SUM or AVG them over any dimension.
- **FACT_SUBMISSION**: stores submission data. The `is_accepted` field (0 or 1) is semi-additive: SUM gives the accepted count, AVG gives the acceptance rate.

6.2.4 Star Schema Diagram

Figure 6.1 shows the star schema for FACT_REVIEW. Both fact tables share the same dimensions, forming what is called a *constellation schema*.

Info

FACT_SUBMISSION does not have a link to DIM_AFFILIATION because submissions are not tied to a specific reviewer. The two fact tables share the other four dimensions, forming a *constellation schema* (also called galaxy schema).

6.3 ETL Process

Data is moved from the operational database to the DW through an ETL (Extract, Transform, Load) process:

1. **Extract**: read new or updated rows from the OLTP tables (Conference, Article, Review, Organizer).

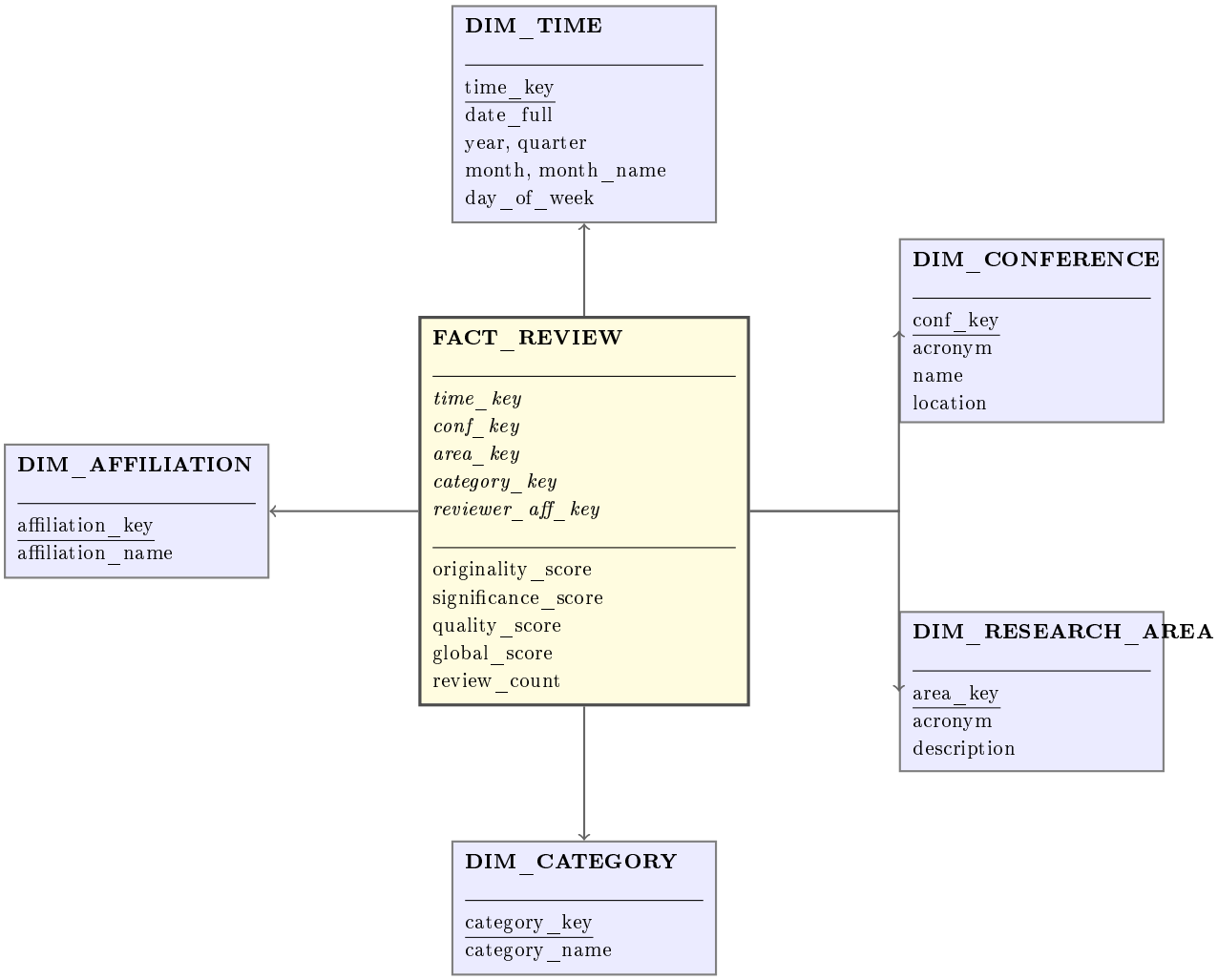

Figure 6.1: Star Schema for FACT_REVIEW.

2. **Transform**: map natural keys to surrogate keys, compute derived fields like `is_accepted`, and extract time components from dates.
3. **Load**: insert the transformed rows into the fact and dimension tables.

Info

The ETL runs as a nightly batch job. Real-time updates are not needed because analytical queries are typically run on a daily or weekly basis.

6.4 OLAP Operations

With the star schema in place, analysts can perform standard OLAP operations:

Table 6.5: Standard OLAP operations.

Operation	What it does	Example
Roll-up	Go from detail to summary.	Monthly → yearly avg score
Drill-down	Go from summary to detail.	Conference → category level
Slice	Filter on one dimension.	Only year 2025
Dice	Filter on multiple dimensions.	Year 2025, location “Bari”
Pivot	Swap rows and columns.	Rows = area, Columns = year

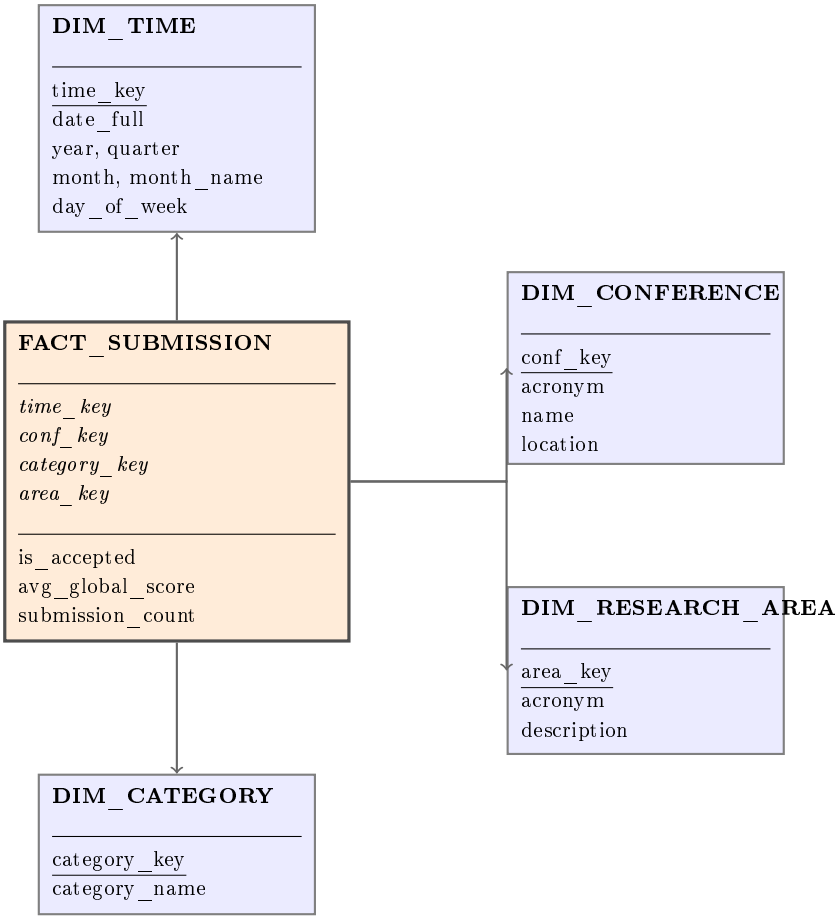

Figure 6.2: Star Schema for FACT_SUBMISSION.