IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

CIRRS: A Relational Database Approach for Managing the Cyber Incident Life Cycle

Om Tushar Johari, Atharva Dattatray Hajare, Aditya Pramod Jadhav, Omkar Vithoba Maan
Guided by: Dr. Mrunal K. Pathak
Department of Information Technology
AISSMS Institute of Information Technology (AISSMS IOIT), Pune, Maharashtra, India

Abstract: *Small organisations and educational institutions often record cyber incidents through emails, phone calls and spreadsheets, which scatters incident details, evidence references and response activities across different places. This paper presents the database design of the Cyber Incident Reporting and Response System (CIRRS), a MySQL relational database that organises incident information from reporting to investigation and resolution. The schema has seven tables: Users, Incidents, Affected_Systems, Evidence, Investigators, Incident_Actions and Resolutions. Primary keys and foreign keys connect the records, while SQL queries, joins and views support retrieval and reporting. The sample database holds 28 inserted rows, including four incidents and one resolution record. The tests show retrieval of related records, an updatable single-table view, a non-updatable aggregate view and the rejection of a delete that would violate referential integrity. The review also found a mismatch between incident status and resolution records: one incident has a resolution record but a status of Under Investigation, and another is marked Resolved without any resolution record. The paper focuses on database design and SQL behaviour, not on automated threat detection.*

Keywords: Relational Database; MySQL; Cyber Incident Management; Entity-Relationship Model; Referential Integrity; SQL Views.

I. INTRODUCTION

Organisations depend on computer systems, networks and online services for daily work, and these systems may face incidents such as malware infections, phishing attempts, unauthorised access and data exposure. Incident-handling and logging guidance [1]–[4] describes the wider process in which incident records are used. Managing one incident requires recording the report, identifying the affected systems, storing evidence details, assigning investigation work, recording actions and documenting the outcome.

When these details are kept separately in emails or spreadsheets, the complete history of an incident is hard to retrieve, and duplicate information and missing links affect report accuracy.

This paper presents CIRRS, a relational database in which the information is stored in separate tables connected by primary-key and foreign-key relationships. The work is a DBMS-focused recordmanagement system. It stores information entered by users and administrators and supports investigation tracking. It does not detect cyberattacks, scan systems for malware, perform intrusion detection or analyse security threats on its own. The paper makes three contributions. First, it gives an entity-relationship model and a seven-table relational schema for the incident life cycle. Second, it tests selected database behaviours with SQL: joins, view updatability and referential integrity. Third, it reports, without correcting it, a mismatch between incident status and resolution records that the sample data exposes. Section 2 surveys related work, Section 3 describes the design, Section 4 reports the experiments and Section 5 concludes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

II. LITERATURE SURVEY

Table 1 summarises the five works that supply the database concepts and incident-handling background used in CIRRS, newest first. The review is limited to the relevance of each source to the project.

Table 1. Literature survey (2025–1970)

Sr. No.	Author(s) and Title	Year	Main contribution	Relevance to CIRRS
1	A. Nelson et al., *Incident Response Recommendations and Considerations for Cybersecurity Risk Management*, NIST SP 80061 Rev. 3 [1]	2025	Gives current recommendations for integrating incident response into cybersecurity risk management.	Provides incidentresponse context; CIRRS implements only record management, not the full framework.
2	R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems* [5]	2016	Explains relational design, keys, normalisation and database constraints.	Guides schema design and the discussion of data redundancy.
3	P. P.-S. Chen, "The EntityRelationship Model—Toward a Unified View of Data" [6]	1976	Introduced ER modelling to represent entities and relationships.	Supports the ER diagram and the identification of the seven entities.
4	E. F. Codd, "A Relational Model of Data for Large Shared Data Banks" [7]	1970	Introduced the relational model for organising data in relations.	Provides the foundation for storing CIRRS records in related tables.
5	Oracle, *MySQL 8.0 Reference Manual* [8]	Onlin e	Documents MySQL features including foreign keys and views.	Supports the interpretation of the referential-integrity and view tests.

These works describe the data model and the incident-response process separately. CIRRS applies the database concepts to the record-keeping part of that process: reports, actions and resolutions. The design rests on ER modelling [6] and the relational model [7], with keys, normalisation, constraints and views as described in standard texts [5], [9]–[11] and the MySQL reference manual [8].

III. METHODOLOGY

Design Approach

The database was built in seven steps: requirement analysis, entity-relationship modelling, relational schema design, table creation, data insertion, query and view development, and testing. Parent records were inserted before dependent records so that foreign-key references pointed to existing rows. Figure 1 shows the entity-relationship diagram, which identifies seven entities, their attributes, keys and relationship cardinalities [12]. The implementation turns these entities into the seven tables of Table 2.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

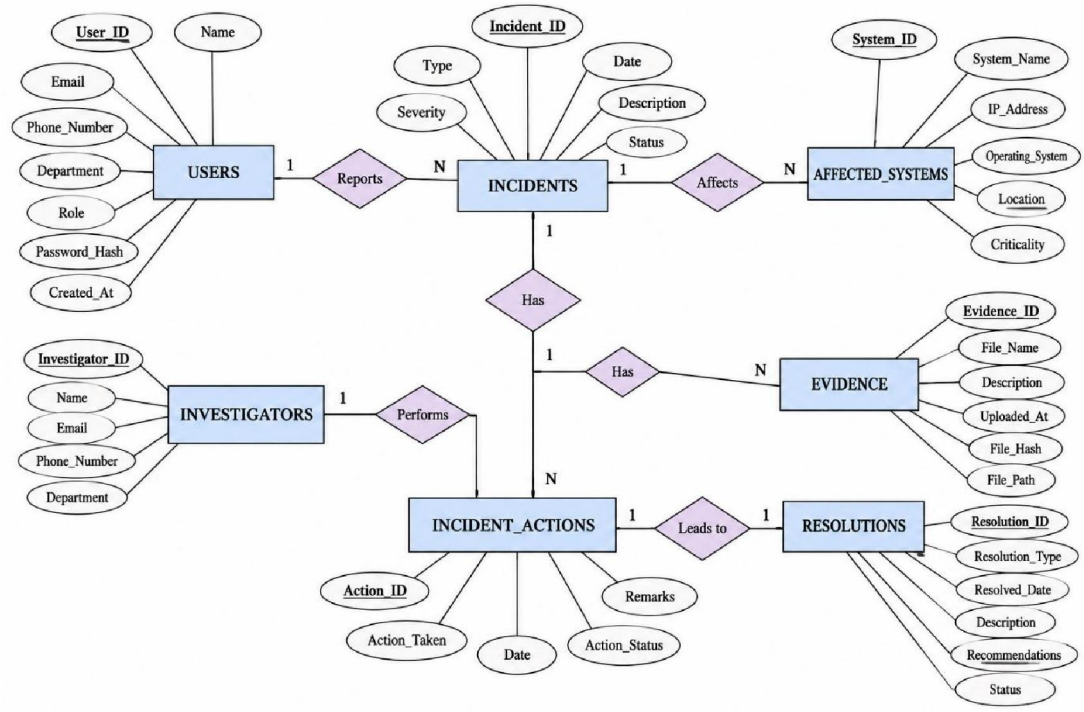


Figure 1. Entity-relationship diagram of CIRRS.

Schema

The schema has seven tables. Table 2 lists them with their primary and foreign keys **Table 2. Tables of the CIRRS schema**

Table	Purpose	Primary key	Foreign key(s)	Rows loaded
Users	Stores user and reporter details	User_ID	—	4
Incidents	Stores incident type, severity, date, description and status	Incident_ID	User_ID → Users	4
Affected_Systems	Stores systems associated with an incident	System_ID	Incident_ID → Incidents	5
Evidence	Stores evidence metadata and file references	Evidence_ID	Incident_ID → Incidents	5
Investigators	Stores investigator details	Investigator_ID	—	4
Incident_Actions	Stores actions performed for an incident	Action_ID	Incident_ID → Incidents; Investigator_ID → Investigators	5
Resolutions	Stores resolution information for an action	Resolution_ID	Action_ID → Incident_Actions	1

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-40756

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

The main design decisions are as follows.

- **Reporter link.** Incidents.User_ID references Users, so every incident is tied to the user who reported it.
- **Evidence metadata.** Evidence stores file paths and descriptive metadata, not the contents of the evidence files.
- **Investigation work.** Incident_Actions references both Incidents and Investigators, so each action is tied to one incident and one investigator.
- **Resolution per action.** Resolutions references Incident_Actions, so a resolution is recorded against a specific action.

Relationships, Constraints and Normalisation

Table 3 lists the relationships between the tables. A foreign key from Resolutions.Action_ID to Incident_Actions.Action_ID ensures that a resolution refers to an existing action. It does not stop several resolution rows from referring to the same action. The one-to-one relationship between Incident_Actions and Resolutions is the intended design, but Action_ID is not declared UNIQUE in the supplied DDL, so a UNIQUE constraint on Resolutions.Action_ID is needed to enforce it.

Table 3. Relationships and cardinality

Relationship	Cardinality	Explanation
Users – Incidents	1 : N	One user can report many incidents; each incident references one user.
Incidents – Affected_Systems	1 : N	One incident can have multiple affected-system records.
Incidents – Evidence	1 : N	One incident can have multiple evidence records.
Incidents – Incident_Actions	1 : N	One incident can have multiple investigation actions.
Investigators – Incident_Actions	1 : N	One investigator can perform multiple actions.
Incident_Actions – Resolutions	Intended 1 : 1; not enforced	A resolution references an action, but Action_ID is not declared UNIQUE in the supplied DDL.

The design separates user, incident, affected-system, evidence, investigator, action and resolution details into distinct relations. This avoids repeating reporter and investigator details in every incident or action row. Primary keys identify individual records, and foreign keys connect dependent records to their parents. The mini-project report describes the schema as being in Third Normal Form (3NF). A formal proof of 1NF, 2NF and 3NF needs the functional dependencies and candidate keys of every relation, and these are not fully documented, so this paper says that the design follows normalisation principles and does not claim a formally proven 3NF schema.

The seven tables are created with CREATE TABLE statements. Each has a primary key, and each dependent table has foreign keys to its parent tables, so a child record cannot refer to a non-existent parent. The DDL printed in the mini-project report has syntax and column-definition inconsistencies. For example, the Users table has a malformed Role declaration, and some columns listed in the data dictionary or used in INSERT statements are missing from the displayed CREATE TABLE statements. This paper describes the logical schema and the test results, and the printed DDL should be checked against the actual MySQL Workbench schema before final publication.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

Views

Two views are defined. High_Severity_Incidents selects the incidents whose severity is High. It is based on a single table without aggregation, so an UPDATE through it can change the matching row in Incidents. Incident_Severity_Summary groups incidents by severity and uses COUNT(*), so it is an aggregate view.

Application Layer

CIRRS also has a browser-based interface built with HTML, CSS and JavaScript, with Python Flask handling backend requests and communicating with MySQL. The mini-project report describes two roles. ADMIN manages accounts, reviews reports, assigns or reassigns investigators and reviews submitted resolutions. USER can report incidents and investigate the incidents assigned to them; investigation is an assigned responsibility and not a separate login role. The web application is only a supporting layer for the database operations. The paper does not claim automated threat detection, malware scanning, intrusion detection or a security analytics engine.

IV. EXPERIMENTATION AND RESULTS

Test Data

The rows loaded in each table are listed in Table 2. The sample contains four incident records. The severity distribution shown by the aggregate view is one Critical, two High and one Medium incident. The affected-system join returns five rows because Incident 101 is associated with two system records, while each of the other incidents has one.

Queries, Joins and Views

SELECT queries display users and incidents, filter incidents by severity, retrieve users by department and sort incidents by date. UPDATE statements change incident status and user department. A DELETE statement for a non-existent user returns zero affected rows. JOIN queries combine Incidents with Users to show reporter details, with Affected_Systems to show impacted devices, and with Investigators and Incident_Actions to show investigation work. An INNER JOIN across Incidents, Incident_Actions Resolutions returns only records that have matching rows in all three tables. Table 4 summarises the tests.

Table 4. Query, view and integrity tests

Test	Observed result	Interpretation
Retrieve all incidents	4 rows returned	Incident records can be retrieved.
Join incidents with reporters	4 rows returned	Reporter details are connected through User_ID.
Join incidents with affected systems	5 rows returned	One incident is associated with two system records.
Join incident actions with investigators	5 action rows documented	Investigator details are linked to actions.
Join incidents, actions and resolutions	1 row returned	Only Action 501 has a matching resolution record.
Test	Observed result	Interpretation
Update through High_Severity_Incidents	Base table row changed	A single-table filtered view is updatable.
Update through Incident_Severity_Summary	Rejected with Error 1288	An aggregate view is not updatable.
Delete a user referenced by an incident	Rejected with Error 1451	The foreign-key constraint prevents the deletion.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

Status and Resolution Consistency

The sample data shows that referential integrity and workflow consistency are different concerns. One resolution record (Resolution_ID 601) is inserted for Action_ID 501, which belongs to Incident 101. Later, an UPDATE through High_Severity_Incidents changes Incident 101 to Under Investigation. Incident 104 is explicitly updated to Resolved, although the sample has no resolution record for its action. Table 5 shows the final documented state. This mismatch is reported as observed and is not corrected in the results.

Table 5. Incident status and resolution records

Incident	Final status	Resolution record	Observation
101	Under Investigation	Yes (Resolution_ID 601)	Resolution exists although the status is not Resolved.
104	Resolved	No	Marked Resolved without a resolution record.

Response-Time Observation

For Incident 101, the report timestamp is 10:30 and the first recorded action (Action_ID 501) is at 11:30, which gives 60 minutes to the first action. The resolution timestamp is 13:00, which gives 150 minutes from report to resolution (Table 6). These figures come from one sample incident and are not systemwide performance measures.

Table 6. Response times for Incident 101

Incident	Reported	First action	Resolution	Minutes to first action	Minutes to resolution
101	2026-10-01 10:30	2026-10-01 11:30	2026-10-01 13:00	60	150

Limitations

- **Small dataset.** The sample has only four incidents and one resolution, so the results are illustrative.
- **DDL inconsistencies.** The displayed DDL has syntax and column-list errors that must be reconciled with the actual database export.
- **Cardinality not enforced.** The intended one-to-one relationship between Incident_Actions and Resolutions has no UNIQUE constraint on Action_ID.
- **Status and resolution can disagree.** They are stored separately, and no tested trigger keeps them in step.
- **Repeated system details.** Affected-system details are stored against individual incidents, so the same real-world system may be entered again for another incident.
- **Duplicated person details.** Users and Investigators are separate entities, so details may be duplicated when one person appears in both roles.
- **No threat detection.** The project manages reported records and does not detect or analyse cyber threats.

V. CONCLUSION

This paper presented CIRRS, a MySQL relational database for organising cyber incident records from reporting through investigation and resolution. The schema has seven tables connected by primary and foreign keys. SQL queries and joins retrieve information across the tables, and views give filtered and aggregate access to incident records. The tests show that a simple view is updatable, that an aggregate view is not, and that a delete which would break referential integrity is rejected.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, November 2026
Impact Factor: 8.2

The data review also found that a valid foreign-key structure does not by itself keep incident status and resolution records in agreement: the final documented updates leave Incidents 101 and 104 inconsistent. The study uses a small academic dataset and shows database functionality, not operational cybersecurity effectiveness.

Future work is to correct and validate the DDL against the final MySQL schema, including all columns used by the INSERT statements; to add CHECK constraints for Severity, Status and other categorical attributes; to add a UNIQUE constraint on Resolutions.Action_ID if the rule is one resolution per action; to add a tested, controlled status-transition mechanism so that resolutions and incident status stay consistent; to introduce an Incident_System junction table so one system can serve several incidents without repeated details; and to test the queries on a larger dataset.

REFERENCES

1. Nelson, S. Rekhi, K. Scarfone and M. Souppaya, *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile*, NIST SP 800-61 Rev. 3, 2025, doi: 10.6028/NIST.SP.800-61r3.
2. International Organization for Standardization, *ISO/IEC 27035-1:2023: Information Technology—Information Security Incident Management—Part 1: Principles and Process*, 2023.
3. K. Kent and M. Souppaya, *Guide to Computer Security Log Management*, NIST Special Publication 800-92, National Institute of Standards and Technology, 2006, doi: 10.6028/NIST.SP.800-92.
4. *OWASP Cheat Sheet Series*. [Online]. Available:
5. https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
6. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2016.
7. P. P.- -relationship model *ACM Transactions on*
8. *Database Systems*, vol. 1, no. 1, pp. 9 36, 1976, doi: 10.1145/320434.320440.
9. *Communications of the ACM*, vol. 13, no. 6, pp. 377 387, 1970, doi: 10.1145/362384.362685.
10. Oracle Corporation, *MySQL 8.0 Reference Manual*. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/
11. Silberschatz, H. F. Korth and S. Sudarshan, *Database System Concepts*, 7th ed. McGraw-Hill, 2019.
12. J. Date, *An Introduction to Database Systems*, 8th ed. Pearson, 2003.
13. H. Garcia-Molina, J. D. Ullman and J. Widom, *Database Systems: The Complete Book*, 2nd ed. Pearson, 2008.
14. Oracle Corporation, *MySQL Workbench Manual: Entity Relationship Diagrams*. [Online]. Available:
15. https://dev.mysql.com/doc/workbench/en/