BCNF∗ – From Normalized- to Star-Schemas and Back Again

Marie Fischer*
Hasso Plattner Institute,
University of Potsdam, Germany
marie.fischer@student.hpi.de

Paul Roessler*
Hasso Plattner Institute,
University of Potsdam, Germany
paul.roessler@student.hpi.de

Paul Sieben*
Hasso Plattner Institute,
University of Potsdam, Germany
paul.sieben@student.hpi.de

Janina Adamcic
Hasso Plattner Institute,
University of Potsdam, Germany
janina.adamcic@student.hpi.de

Christoph Kirchherr
Hasso Plattner Institute,
University of Potsdam, Germany
christoph.kirchherr@student.hpi.de

Tobias Straeubig
Hasso Plattner Institute,
University of Potsdam, Germany
tobias.straeubig@student.hpi.de

Youri Kaminsky
Hasso Plattner Institute,
University of Potsdam, Germany
youri.kaminsky@hpi.de

Felix Naumann
Hasso Plattner Institute,
University of Potsdam, Germany
felix.naumann@hpi.de

ABSTRACT

Data warehouses are the core of many data analysis processes. They contain various database schemas, which are designed and created through schema transformation and integration. These processes are complex and require technical knowledge, which makes them costly and prevents business teams to start new analyses independently. BCNF∗ is a web application that enables users to safely explore valid schema transformations and generate transformation scripts automatically. It can be used for any schema transformation, but is optimized for semi-automatic data warehouse creation through means like a dedicated star schema mode.

CCS CONCEPTS

• **Information systems → Database utilities and tools**.

KEYWORDS

schema management; star schema; normalization

ACM Reference Format:
Marie Fischer, Paul Roessler, Paul Sieben, Janina Adamcic, Christoph Kirchherr, Tobias Straeubig, Youri Kaminsky, and Felix Naumann. 2023. BCNF∗ – From Normalized- to Star-Schemas and Back Again. In *Companion of the 2023 International Conference on Management of Data (SIGMOD-Companion '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555041.3589712

1 SCHEMA MANAGEMENT

Companies often manage data in data warehouses. Despite data lakes gaining popularity in recent years, much structured data is still stored in well-defined database schemas. Organizing the same data in multiple schemas can improve the performance of different workloads.

Businesses tend to store unprocessed and denormalized data in a *landing schema*. This schema offers easy extensibility and low complexity. It comes at the cost of data duplication and a lack of consistency due to possible data anomalies. Structuring data in normalized *core schemas* reduces redundancy, improves data integrity and requires less physical space. However, performing data analysis workloads (OLAP) on normalized data usually requires multiple costly joins. To overcome downsides of core schemas for analysis, data is restructured into *star schemas*, consisting of fact and dimension tables [4]. Often, there are multiple workload-specific schemas for different analyses. Data analysis tools rely on such specialized schemas to perform analytical queries effectively [9, 10].

As different schemas offer individual benefits, organizations want to efficiently transform schemas back and forth. These transformations demand high manual effort and a good understanding of data profiling techniques to identify and utilize data dependencies. Furthermore, knowledge about the underlying business context is essential to assess data layouts and grasp domain-specific design choices. Lastly, familiarity with data processing languages, such as SQL, is required to perform the desired transformations and build necessary data pipelines. Hence, data analysts, domain experts, and data engineers have to work collaboratively to transform schemas back and forth.

We present BCNF∗, an interactive open-source web application[†] for flexible schema management and integration. With it, users can easily transform schemas from normalized forms to star (∗) schemas and back. In BCNF∗, we normalize schemas using the popular Boyce-Codd normal form (BCNF); other normal forms can be added to BCNF∗ in the future.

2 BCNF∗ OVERVIEW

BCNF∗ is a web application that facilitates the end-to-end process of schema transformation. Additional dedicated modes offer a variety of tools to simplify the creation of star schemas and the integration of multiple schemas.

*Authors contributed equally to this research

SIGMOD-Companion '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-9507-6/23/06.
https://doi.org/10.1145/3555041.3589712

[†]https://github.com/SchweizerischeBundesbahnen/BCNFStar

2.1 Schema Editing Mode

Schema editing in BCNF∗ is supported by data dependencies, namely functional dependencies (FDs) and inclusion dependencies (INDs). In this way, we ensure that all transformations preserve the integrity of the schema. The usual workflow with BCNF∗ is divided into three steps:

(1) Configuration and execution of dependency discovery algorithms on the data
(2) Schema editing using the calculated dependencies
(3) Persistence of the final schema by applying the necessary transformations on the data

Dependency recognition is performed by algorithms implemented in the Metanome data profiling framework [5]. The default algorithms used for FD- and IND-discovery are HyFD [7] and BINDER [6], respectively. We allow the configuration of algorithm parameters (e.g., maximum number of columns for the left-hand side of FDs).

After this phase, the FDs and INDs that exist on the schema are known and can be used for integrity preserving schema transformation as follows. BCNF∗ can be easily extended to support more algorithms that run in the Metanome framework.

Schema editing in BCNF∗ is primarily supported in the form of semi-automatic normalizing, joining and unioning of tables. Based on a previously discovered and automatically suggested FD, a table can be normalized. The columns of the left-hand side and right-hand side of the FD form a new table. This new table is referenced by the initial table via a foreign key (FK) on the columns of the left-hand side. The columns of the right-hand side can be deleted from the initial table to complete the normalization.

INDs occur if the distinct values of one column set A are included in the distinct values of a column set B. If B forms a candidate key in its table, the IND may represent a FK relationship between the two tables. BCNF∗ allows identifying such INDs as an actual FK, and based on such FKs, two tables can be joined.

As a matter of fact, datasets tend to contain a lot of FDs that should not be used for normalization or inclusion dependencies that do not represent an actual FK relationship. Thus, BCNF∗ offers various rankings helping the user distinguish randomly occurring dependencies from correct FDs and INDs, including rankings found in [1, 8, 12–14]. The rankings can be adapted by the user, i.e., depending on the data, a different weighting of different ranking algorithms could provide the best results.

Due to incorrect tuples in real-world datasets, FDs and INDs that should hold are not valid. Therefore, a user cannot find such dependencies when editing the schema in the transformation proposals. In this case, BCNF∗ provides the possibility to suggest certain dependencies manually by the user and show violating tuples.

Further functionalities like adding surrogate keys, table and column renaming, or the deletion of FKs, tables and columns help the user to reach the desired schema.

Persistence of the final schema is accomplished through the generation and execution of SQL-statements that perform the user's transformations. The new tables are created and filled with data, key constraints are established and surrogate keys generated. Currently, BCNF∗ supports PostgreSQL and SQL Server dialects; extending BCNF∗ to support other DBMSs is easy. Schema design is also a collaborative and iterative task. Therefore, BCNF∗ allows data scientists to save their progress in a JSON-based format. The file can be shared with colleagues to collaboratively edit the schema, or reloaded to continue schema editing later.

2.2 Star Schema Mode

Structuring data in a star schema is of great practical importance for organizations. BCNF∗ offers a dedicated mode to support the transformation of (normalized) data into workload-specific star schemas. To this end, BCNF∗ automatically identifies and highlights tables as *facts* and *dimensions*: Facts are tables that are not referenced by any other table. Dimensions, on the other hand, are tables that are referenced by other tables and contain additional information. If dimensions are referenced in the fact table, the tables are *direct dimensions*, if they are transitively referenced by other dimensions we call them *indirect dimensions*, as shown in Figure 1.

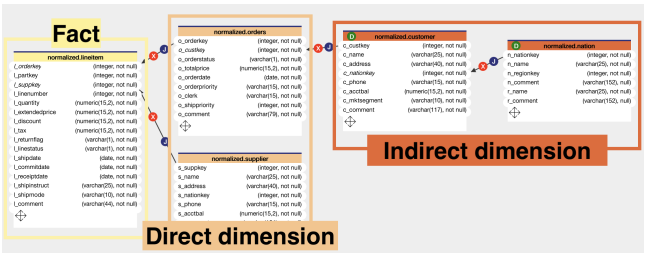

Figure 1: Facts, direct and indirect dimensions

BCNF∗ also calculates the *fact potential* for non-fact tables based on the Connection Topology Value [11]. If this value surpasses the schema-specific threshold from [2], the user can manually specify such tables as facts as well. For indirect dimensions that are linked to facts via several other tables, the star schema mode allows creating the corresponding direct references automatically.

Additionally, we filter and do not display FKs that are probably irrelevant for star schema design. For instance, FKs among (indirect) dimension tables should not be persisted as a database constraint. For better illustration, Figure 2 shows the filtering of the FKs in the schema editing and star schema mode. FKs displayed in only one of the modes are highlighted in green. Activating FKs manually displays them again and ensures their creation at schema persistence. In general, FKs are only persisted when visible (WYSIWYG).

2.3 Schema Integration Mode

The schema integration mode is intended for users who want to include a new data source / schema by integrating it with the existing schema. BCNF∗ guides users by splitting the process into three phases: the alignment, the fusion, and the persistence phase.

At the start of the alignment phase, equivalent columns between the two schemas are searched with the COMA-3.0 schema matcher [3]. Afterward, the user transforms the schemas so that these corresponding columns build corresponding tables using a specialized compare-view, as shown in Figure 3. In this example, COMA-3.0 identified that the columns of the denormalized table (left) match three separate tables of the normalized schema (right).

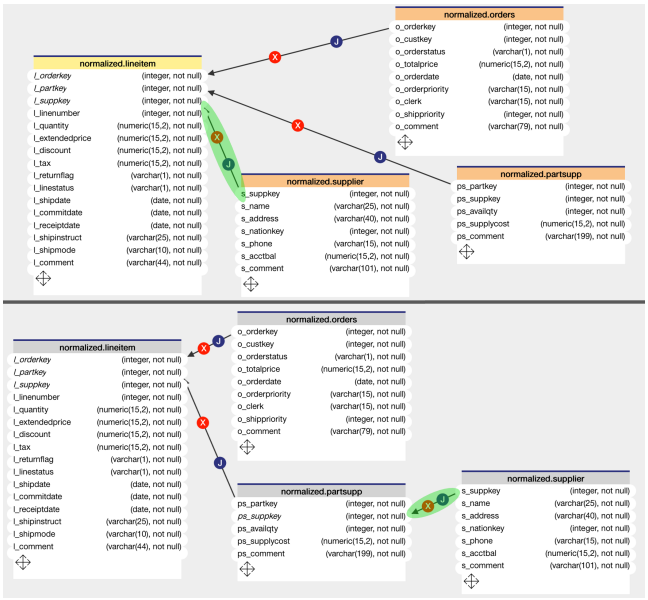

Figure 2: FKs displayed in schema editing mode (above) and star mode (below)

To align the schemas, the user can either normalize the left table based on FDs or join the right tables based on FKs or INDs.

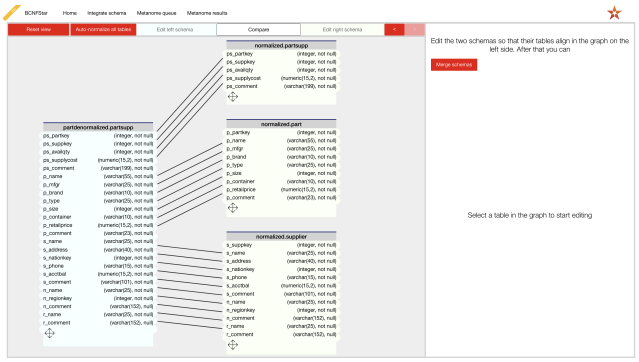

Figure 3: Compare view of the integration mode

In the fusion phase, equivalent tables are unioned to form one resulting schema. The persistence is done as in the schema editing mode of BCNF∗. Here, only the integrated schema remains. Finishing touches can be added, and SQL scripts generated to execute the integration on the database.

3 DEMONSTRATION SCENARIOS

We demonstrate two typical workflows with BCNF∗. In our demonstration scenario, a data scientist is interested in transforming the schema of a denormalized TPC-H dataset*. Firstly, she wants to retrieve the original normalized TPC-H schema, to avoid data anomalies and require less memory. In a second step, she would like to create several analyses to answer business questions, such as (1) Which line items are most frequently purchased by customers from China or (2) Which country imports the most parts for each line item respectively? To execute the requests for these analyses efficiently, she creates a star schema.

*https://www.tpc.org/tpch/

3.1 Normalizing a Schema

In our first part of the scenario, the data scientist wants to normalize a schema. She selects the denormalized tables that must be normalized and configures the algorithms: All available memory should be used and FDs shall be limited to two left-hand side columns. Also, the IND calculation can be restricted to INDs with a maximum of two columns in this scenario. This speeds up the FD/IND calculation compared to the default settings.

After automatically discovering the FDs and INDs, the data scientist starts working on the transformations. Figure 4 shows the three components of the GUI: (1) the graph, which displays the tables, their columns and foreign key (FK) constraints between tables, (2) the sidebar, containing the transformation menus and (3) the menu bar, which allows switching between normal and star schema modes (see Sections 2.1 and 2.2).

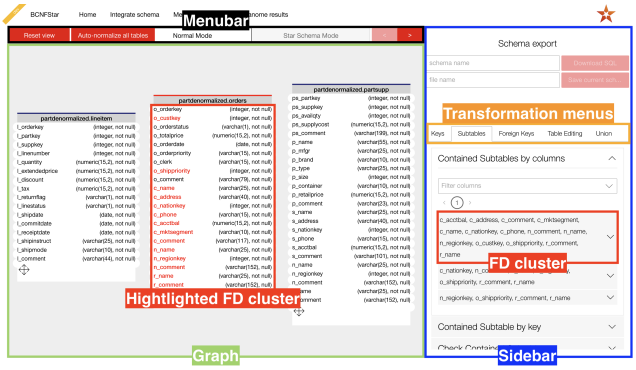

Figure 4: Schema editing mode

The data scientist selects a table to open the transformation view in the sidebar, which provides the functionality for split, join and union tasks. The *Subtables* tab displays FDs in clusters. These clusters group FDs, which would create the same table when splitting, just with a different primary key. For better illustration, the columns of a cluster are highlighted when the mouse is moved over the corresponding cluster.

Based on the FD ranking, she decides to normalize the orders table by separating customer information with the FD o_custkey → c_name, c_address, n_regionkey, n_name, ... Before the split based on this FD, as described in Section 2, is automatically executed, she can manually specify that certain columns are not to be split out.

She additionally searches for FDs containing the n_nationkey to normalize the table further by creating a table with the nation information. She repeats similar steps for all three source tables and ends up with three normalized but yet unrelated schema components.

To connect those individual components, BCNF∗ offers the previously calculated INDs that are displayed in the tab *Foreign Keys*. By selecting the corresponding IND, the data scientist creates one

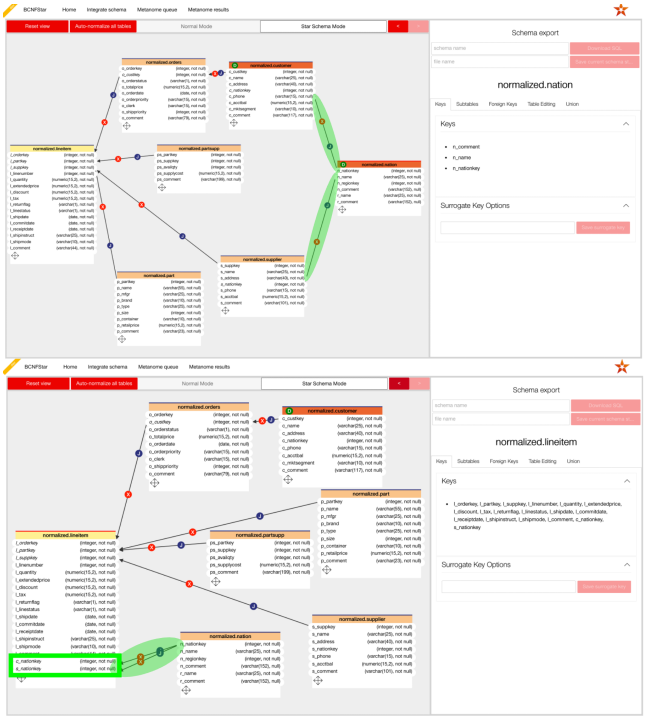

Figure 5: Direct dimension creation: before and after

FK from l_orderkey in lineitem referencing o_orderkey in orders and another FK from l_partkey, l_suppkey in lineitem referencing ps_partkey, ps_suppkey in partsupp.

BCNF∗ lets the data scientist persist the aforementioned schema transformations by generating a migration file. The file contains the SQL-statements to create the new tables, insert the (joined and/or unioned) source data into them and establish (surrogate-) key constraints. The SQL-code for the migration file can contain several hundred lines, even for a few transformations.

3.2 Creating a Star Schema

The second part of the scenario shows how BCNF∗ can be used to create a workload-specific star schema from an existing normalized schema. The data scientist loads the normalized schema and switches to the star schema mode. There, the tables are automatically highlighted as facts (yellow), dimensions (orange) and indirect dimensions (red). As the analyses do not require region information, the data scientist can delete the corresponding table.

In order to create the star schema, the data scientist connects the two remaining *indirect dimensions* to the *fact*, making the former *dimensions*. With the default operations discussed previously, this would require numerous transformations. However, the star schema mode allows creating such a reference immediately.

In some cases, an *indirect dimension* may be referenced by multiple *dimensions*: In Figure 5, the supplier and the customer tables are dimensions of lineitem. Both reference the nation table. The final analyses rely on the customer's and supplier's nations. Thus, the nation table has to be linked twice to the fact table. To realize the schema persistence in the end, BCNF∗ remembers the initial schema and required joins that allow those references.

After finishing the star schema creation, the schema can be persisted as described before, and ultimately imported into any analysis tool (e.g., PowerBI) to create the analysis reports.

4 CONCLUSION

We presented BCNF∗, a web application for flexible schema management and data integration. Through an upstream analysis of the data, users are provided with numerous suggestions for schema transformations. This speeds up the identification of data relationships and allows a semi-automatic editing of schemas, easily moving between denormalized, normalized and star schemas.

The user is additionally guided in creating workload-specific star schemas or integrating multiple data sources. With the interactive GUI, BCNF∗ facilitates the end-to-end process of complex schema transformations, even for those, that are not familiar with the underlying schema or lack the technical skills.

ACKNOWLEDGMENTS

We are grateful to Simon Staudenmayer, Andreas Gehri, René Helg, and Germann Phillip of SBB Cargo AG, for their support.

REFERENCES

[1] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2016. On the Discovery of Relaxed Functional Dependencies. In *Proceedings of the International Database Engineering & Applications Symposium (IDEAS)*. 53–61. https://doi.org/10.1145/2938503.2938519

[2] Richard D Christie. 2002. Statistical methods of classifying major event days in distribution systems. In *IEEE Power Engineering Society Summer Meeting*, Vol. 2. IEEE, 639–642.

[3] Hong-Hai Do and Erhard Rahm. 2002. COMA — A system for flexible combination of schema matching approaches. In *Proceedings of the International Conference on Very Large Databases (VLDB)*. Morgan Kaufmann, San Francisco, 610–621. https://doi.org/10.1016/B978-155860869-6/50060-3

[4] Ralph Kimball and Margy Ross. 2011. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons.

[5] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. 2015. Data Profiling with Metanome. *PVLDB* 8, 12 (2015), 1860–1863. https://doi.org/10.14778/2824032.2824086

[6] Thorsten Papenbrock, Sebastian Kruse, Jorge-Arnulfo Quiané-Ruiz, and Felix Naumann. 2015. Divide & conquer-based inclusion dependency discovery. *PVLDB* 8, 7 (2015), 774–785. https://doi.org/10.14778/2752939.2752946

[7] Thorsten Papenbrock and Felix Naumann. 2016. A hybrid approach to functional dependency discovery. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 821–833. https://doi.org/10.1145/2882903.2915203

[8] Thorsten Papenbrock and Felix Naumann. 2017. Data-driven Schema Normalization.. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 342–353. https://doi.org/10.5441/002/edbt.2017.31

[9] PowerBI 2022. https://powerbi.microsoft.com/de-de/. [Online; accessed 09-December-2022].

[10] PowerBI 2022. Understand star schema and the importance for Power BI. https://learn.microsoft.com/en-us/power-bi/guidance/star-schema. [Online; accessed 09-December-2022].

[11] Il Yeol Song, Ritu Khare, and Bing Dai. 2007. SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP (DOLAP)*. 9–16.

[12] Ziheng Wei, Sven Hartmann, and Sebastian Link. 2020. Discovery algorithms for embedded functional dependencies. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 833–843. https://doi.org/10.1145/3318464.3389786

[13] Ziheng Wei, Sven Hartmann, and Sebastian Link. 2021. Algorithms for the discovery of embedded functional dependencies. *VLDB Journal* 30, 6 (2021), 1069–1093. https://doi.org/10.1007/s00778-021-00684-3

[14] Ziheng Wei and Sebastian Link. 2019. Discovery and ranking of functional dependencies. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 1526–1537. https://doi.org/10.1109/ICDE.2019.00137