Database Query Execution Through Virtual Reality

Logan T. Bateman, Marc A. Butler, and Germán H. Alférez(✉)

School of Computing, Southern Adventist University,
PO Box 370, Collegedale, TN 37315-0370, USA
{lbateman,marcbutler,harveya}@southern.edu

Abstract. Building database queries often requires technical knowledge of a query language. However, company employees (outside of software development, generally) may not have the expertise to accurately construct and execute database queries. Moreover, in the era of Big Data, we believe that it is important to count on more flexible interfaces that may allow users interact with data in a more natural and immersive way. Our contribution is to utilize virtual reality (VR) to construct database queries and execute them on relational databases. Using natural hand or controller gestures, this approach seeks to provide easy access to build and visualize database queries. This is a first approach towards more flexible interfaces to interact with data.

Keywords: Relational database · Virtual reality · Natural human interface

1 Introduction

Industry's sustained operation is contingent on the data to be maintained, whether that data regards customers, business metrics, analytics, or simply product inventory. The large number of databases and increasing size of each, prompts for the design of new and innovative ways for viewing and manipulating this data. Currently, database administrators have many options, but there are fewer options for those who use database systems less frequently and do not have a full working knowledge of database query languages such as the Structured Query Language (SQL).

Existing tools for data manipulation include DBeaver[1], MySQL Workbench[2], and phpMyAdmin[3]. While helpful to the expert, their 2D interfaces are often highly technical in nature, providing tools to access diagrams, create scripts, and procure data. For any non-trivial operation, however, technical expertise is an absolute requirement.

[1] https://dbeaver.io/.
[2] https://www.mysql.com/products/workbench/.
[3] https://www.phpmyadmin.net/.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
K. Arai (Ed.): FTC 2022, LNNS 560, pp. 619–633, 2023.
https://doi.org/10.1007/978-3-031-18458-1_42

The emergence of VR provides a new way of visualizing and interacting with data and systems. Its application can be seen in many fields including education, business, and the sciences. For example in [1], cardiologists can improve their accuracy through visualizations designed for medical operations. Specifically, the approach in [1] relies on VR to map objects in 3D to non-tangible concepts. Positional/orientation tracking controllers can be mapped to the 3D space to allow novel computer interactions, which until recently required use of a mouse, keyboard, or standard controllers.

This paper proposes a solution for database query generation using VR interactions to replace the typical method for creating database queries (queries by means of typing SQL with a keyboard). A variety of state-of-the-art software tools/libraries as well as VR hardware (headset and controllers) are used. For instance, some tools provide the ability to support gestures such as hand and force grabbing, pinching, pulling and pushing, and the motion of the observer's head. Controller mapping and rigging of virtual appendages are already available through existing toolkits. By means of presenting a solution based on the latest VR technologies, we intend to move further from the state of the art, which has focused on data visualization, not on the execution of database queries. Moreover, we believe there is a need for exploring new VR technologies, both in software and hardware, which may allow more intuitive user interactions with databases in the future.

The hope is that in the future users will be able to organically query data by means of more natural interfaces, in this case manipulation of 3D geometric primitives using the known advancements in VR technology. This research work is limited to explain the technologies that can be combined to achieve database queries through VR. A simple example shows the applicability of our approach at applying select and join operations between tables in a simple databse. In the example, basic database querying is necessary. In future work, we expect to evaluate our technological approach with a group of users in real scenarios.

This paper is organized as follows: Sect. 2 dives into state of the art technology that provides ground-breaking solutions regarding databases, VR interaction, and game engines. Section 3 presents the theoretical framework which utilizes state of the art tools and technologies, along with revolutionary, cutting-edge technology, not frequently employed before now. Section 4 presents the methodology to explore the process of finding and choosing hardware and libraries. This section also covers the implementation of user interaction, a database solution, and a toolkit for our chosen game engine. Section 5 outlines the result of the research conducted, namely the tool created. Section 6 presents conclusions and future work.

2 State of the Art

This section explores VR approaches and prototypes for database visualization and queries as a means to achieve natural human interface to interact with data.

VR has been used for information retrieval in different fields. For instance, in [2], the authors present a very complete literature review in this topic. However,

there are few research works in the area of integrating VR and databases. These research works originated from the 90s [3]. Although these approaches paved the way towards more intuitive forms of interacting with databases, they did not mature with the current progress of VR, for instance, with new VR devices. Therefore, there is a need for experimenting with new VR technologies to help users have a more intuitive interaction with data.

In the area of databases, a similar approach to ours is presented in [4]. In this research work, the author presents an approach for visualizing, navigating, and conveying database model information using a web browser by means of the WebVR technology. However, this approach focuses on visualizing the structure of the database, not on executing queries on the database.

Another study focuses on querying data through VR [5]. In this research work, the authors present an approach called Queryball for querying in immersive VR systems. Queryballs are translucent balls that handle search conditions, applied only to the virtual objects inside of the Queryball. Users operate Queryballs by means of a stylus pencil and a virtual board, which are outdated ways of interacting with virtual environments.

As shown in [6], it is essential to provide a pleasing user experience that establishes a strong computer-human relationship. To this end, the authors developed a tool to provide an intuitive and simple 3D virtual environment that is easily navigable and comfortable for extended usage. In another research work [7], the authors show an interesting example, termed 'visual linking', which makes use of 3D objects in VR to represent concepts. In their use of visual linking, graphs in 3D space contained points on multiple axis which directly map to a 3D object. This type of 3D representation is useful in encouraging natural human interaction; no click or key press is required.

The aforementioned approaches try to move from traditional 2D contexts to 3D scenarios. For instance, DbVisualizer[4], a database visualization software with query building functionality, allows the building of queries in a graphical manner. This software, for example, allows the execution of a join by clicking and dragging a column from one table to a column in the other. While DbVisualizer provides a 2D interface for designing queries with a mouse and a keyboard, it does not provide a highly immersive experience for the user. Traditional graphical software like this is based on the Window, Icon, Menu, Pointer (WIMP) paradigm, but does not allow 3D interaction, reducing the ability to interact naturally [8].

Current research on database visualization typically involves a VR application with Unity Engine as shown in [4]. The research work in [4] is highly related to ours as visualization of table structure is an important part of designing database queries. Further work should be focused on natural query building.

EntangleVR [9] is a system that abstracts knowledge of quantum computing and quantum physics through virtual environments. Much like EntangleVR provides a visual programming interface for topics in quantum entanglement, this research seeks to provide a visual programming interface that is simple and intuitive for the construction of SQL queries.

[4] https://www.dbvis.com/.

Another application called UrbanVR for urban design planning provides to users a sense of physically being present in the virtual environment [10]. They can then perform shading and visibility analysis. Also, their research includes multiple gestures for selection, translation, rotation, and scaling of objects. Similar variations on these gestures are crucial in this research to ensure a positive user experience.

A goal concerning user interaction is to reduce the amount of user input required to obtain a useful output. This technique can be referred to as Direct Manipulation because the user does not have to type commands or take many steps in a GUI to receive feedback. Rather, a user can naturally manipulate a virtual environment, producing significant output as mentioned in [11].

Prior research work has compared the use of a graphical user interface (GUI) versus a natural user interface (NUI) for delivering interactive narratives, as shown in [12]. In this work, researchers sought to discover if a GUI or NUI was more comfortable for users in 3D. It exemplifies the importance of user 'presence', meaning that the user feels like they are physically present and can interact with objects naturally. This concept supports the idea that physical presence can allow query building to be more natural.

Finally, in [13] the authors describe a prototype of a distributed multimodal virtual environment used for the visualization and query of complex data. However, implementations details, specifically in the area of query execution, are not given.

The above-mentioned tools and techniques show interesting uses of VR for visualization and tools for query/manipulation of databases. VR has even been used to convey database structure. However, most research works on VR and databases tend to focus on data visualization, not on data queries. The few approaches that deal with query execution on databases are outdated or not present sufficient technical details. This research aims to tackle and introduce a way to start to fill this gap by means of using state-of-the-art VR technologies.

3 Theoretical Framework

The described solution of creating a natural user interface for building database queries can be seen in the overview in Fig. 1. At the core of the model is VR. It is from this key concept in which a game engine is used for 3D representation of a system allowing database access. The 3D environment rendered by the game engine may be manipulated via physical input devices such as interactive controllers and a VR headset.

This research work aims to develop an interface for interacting with a SQL Relational Database Management System (RDBMS). One such system is SQLite, an in-memory or single-file database management system (DBMS) that can run on a multitude of devices, from embedded devices to desktop applications[5]. It aims to be very small and self-contained for portability and ease of use in all

[5] https://www.sqlite.org/index.html.

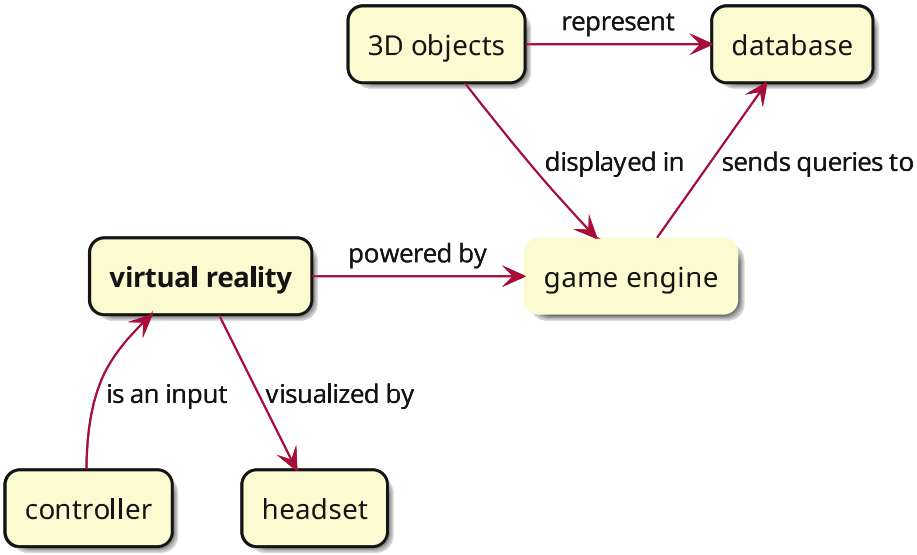

Fig. 1. Conceptual Map

environments. In this research work, SQLite is used because no dedicated DBMS server is required and integration is fairly straightforward in multiple languages.

In VR 3D space, user interaction typically uses tracking of head and hands as input. An Oculus Rift-S is used in this research work, utilizing controllers that track the position/orientation of hands as well as providing physical buttons and analog sticks for finger input.

To facilitate 3D rendering and user input, the Unity Engine[6] was used. Unity is a flexible platform and game engine for creating games and other interactive applications. Unity uses GameObjects to store objects and state. To these objects, developers can attach Components, which can be anything from a script (C#) to a Mesh Collider. Physics is included in the game engine natively, allowing for simulation of physical actions like the pressing of a button, falling with the force of gravity, or momentum from colliding objects. This can help make concepts like database tables more tangible.

Instead of using touch controller input directly, XR Interaction Toolkit can be used. This is a Unity plugin/high-level library for creating AR and VR experiences[7]. This tool provides two different methods for getting user input, action-based and device-based input. Action-based input is where the developer will choose functions or operations for each action regardless of device model. Device-based input uses input of controllers directly without intermediary actions. An

[6] https://unity3d.com.

[7] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/index.html.

action-based approach makes sense to use if the application is being developed to work with multiple VR platforms (Oculus Rift, HTC Vive, Valve Index, etc.).

To create models for use in 3D space, Blender[8], a free and open source 3D modeling and animation software can be used. Models created with Blender can be imported into Unity or other game engines. This is necessary because Unity engine does not provide any mesh modeling capabilities; only scaling, rotation, and position can be modified.

4 Methodology

The process of creating a VR tool to query a database has multiple components. Hardware must be selected, as well as an engine/API for interfacing with the hardware and rendering a 3D environment. This section covers the steps in the development of this tool.

4.1 Choosing XR Hardware

Hardware had to be chosen to interact with and use the proposed VR database interaction tool. Several options were considered including Microsoft Hololens, Oculus Rift-S, and others. Availability and convenience ended up being the deciding factor in choosing to use an Oculus Rift-S for this research work. The Oculus Rift-S is a headset with inside-out position tracking and rotation tracking with six degrees of freedom (6DoF) as well as a screen with a resolution of 2560×1440 at 80 Hz refresh rate. Six degrees of freedom refers to the ability to measure movement along x, y, and z while also measuring rotation by roll, yaw, and pitch. Along with the headset, Oculus Touch controllers were used, which also support 6DoF for intricate movement and rotation in 3D space. The selected hardware was found to be sufficient for the tool and can be seen in Fig. 2.

4.2 Finding a 3D/VR Engine

After choosing VR hardware, a suitable 3D engine had to be selected. The platforms considered were Unity Engine, Unreal Engine, and Godot Engine. Each of these engines have their own unique advantages and disadvantages.

Although Unreal Engine is very powerful, it does not have as large of a community with tutorials and tools. Also, although Unreal Engine provides a C++ API, it does not provide C# bindings without a third party integration like UnrealCLR[9]. The Godot engine was also considered an option and has major positive features such as being open source and having C# scripting ability. However, Godot was not chosen because of its lack of extensive VR tooling/libraries.

Ultimately, the Unity Engine was chosen for its ease of use, extensive VR support, and C# API. Additionally, the tools available for Unity such as the XR Interaction Toolkit made prototyping in VR more convenient.

[8] https://blender.org/.
[9] https://github.com/nxrighthere/UnrealCLR.

Fig. 2. Operation of Rift-S with Oculus Touch Controllers

4.3 Finding a DBMS

With hardware and a 3D engine, the other component required for the tool is a database to run the created queries on. Thus, a suitable DBMS had to be chosen. Many DBMS exist including but not limited to MariaDB, MySQL, PostgreSQL, Microsoft SQL, SQLite, etc. Each of these would work for this tool, but SQLite was chosen because it requires no dedicated database server and can use a single .sqlite file. Also, SQLite has a C# connector which is readily available: Mono.Data.SqliteClient[10]. Other databases are more feature rich and often more performant (concurrent access, network connections, very large datasets, etc.)[11], but not as convenient for the goal of this paper. Also, Microsoft SQL Server could be a good option as C# was being used for scripting. However, creating a connection proved to be more difficult than expected, likely because Unity supports Mono[12], which is not exactly the same at DotNet Framework/Core.

5 Results

The resulting tool provides a way to query a SQLite database in VR. The implemented SQL operations were SELECT and JOIN. A query using these operations is built through various physical actions. When cubes which represent relations are brought together in 3D space, the generated query contains a JOIN (specifically a Cartesian product), where every row in one table is paired with every row in the other table. The query is executed when the user moves his or her right hand near one of the cubes. This triggers tabular output to be displayed on the wall in front of the user. Figure 3 exemplifies the architecture of the tool.

[10] https://www.mono-project.com/docs/database-access/providers/sqlite/.
[11] https://www.sqlite.org/whentouse.html.
[12] https://www.mono-project.com/.

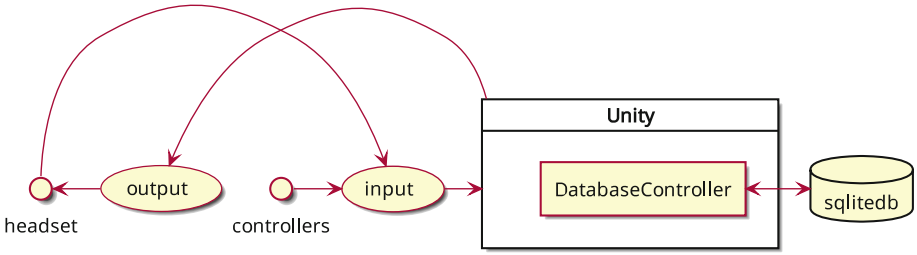

Fig. 3. Architecture Diagram of the Tool

The dataset used to demonstrate the functionality of the tool is about an inventory relation which describes the quantity of foods. There is a zero-to-many relationship between specific foods and records of inventory. This means that a particular food can have no inventory or inventory records with different expiry dates and quantities. The exact fields can be observed in Fig. 4. A third relation named food_type is not included here to improve readability in the explanation of the approach in the following sections.

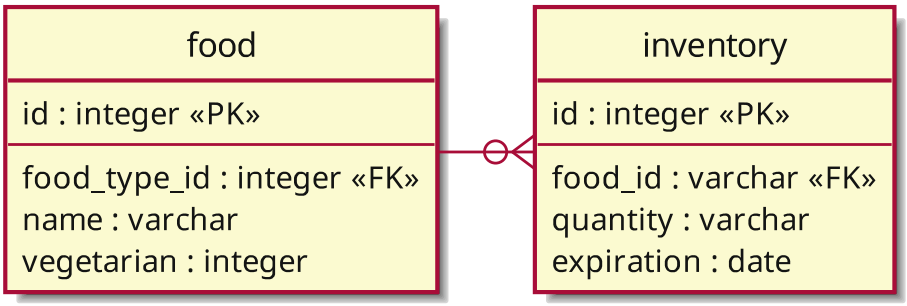

Fig. 4. Example Data Tables

5.1 Relation Cubes

The main component of the tool are cubes, which represent relationships or mappings between each table in the database and the virtual environment (see Fig. 5). In the example, there are two relation cubes in a 3D Unity game scene, one for each of the two tables presented in Fig. 4. The user can see them in a 3D space when the tool starts and a VR headset is being worn. These cubes were generated by a DatabaseController script which assigns a relation to each cube created.

The relation cube has a setter for table name that is in charge of properly showing the corresponding text (name) on/near the cube. The relation cube contains a single Unity canvas object, which allows UI elements to be displayed

Fig. 5. Relation cubes

(e.g. buttons, rasterized text, scroll rects, etc.). Within each canvas GameObject is one or more TextMeshPro GameObjects, allowing text to be displayed. The text can be changed via an attribute on the cube. Lines 5–6 in Listing 1 is getting the text mesh (label) on the cube and setting its text content, which updates in 3D space.

```
1  class TableCube {
2    # .. other code .. #
3    name {
4      set {
5        textMesh = getTextMesh
6        textMesh.text = value
7      }
8    }
9  }
```

Listing 1. Relation cube setter

Queried relation attributes from the database have been placed on the relation cube at various positions. There may be several TextMeshPro components on a single GameObject, which hold the text of each relation attribute. A brief example of how resizing and setting the attributes for a given relation cube can be seen in the pseudocode in Listing 2. Line 2 computes a vertical scale for the relation cube based on the number of attributes. Lines 3–6 distribute the vertical positions of text meshes, which get placed on the canvas.

```
# adding attributes to relation cube
cube_y_scale *= count(attribs) / SCALING_FACTOR
for index, attrib in attributes:
  textObject.y = offset + PLACEMENT_FACTOR * index
  textObject.text = attrib.name
  add textObject to canvas
```

Listing 2. Relation cube setup

5.2 Display of Select Output

The user operates within a hollow cube, previously created in Blender. The cube is akin to a movie theater in that it displays information on one of its one walls, via a canvas and text mesh. The user's point of view (POV) is larger than what is shown in Fig. 5, because the user is wearing a headset. Select query output relating to a relation cube or a combination of relation cubes can be seen in Fig. 6. In Listing 3, line 1 creates a SQL query, inserting the name of the relation. Lines 2–6 extract the output and place it in a string with tabs and newlines. Finally, in line 7, the content of the text mesh on the wall is set.

Fig. 6. Relation Cube with Query Data

```
genSqlText = ``SELECT * FROM {tableName}"
output = ``"
for row in results:
  for value in row:
    output += value + ``\t"
```

```
    output += ``\n"
  textMesh.text = output
```

Listing 3. Query execution

Both the canvas on the wall and its child TextMeshPro GameObject are scaled during design time (before running) to comfortably fit query output. The text within the single text mesh contains tabs and newlines to create a tabular display of the query output.

5.3 User Input

The query output is not displayed without user engagement. To this end, the user is able to move their hands in the interactive simulation shown in Fig. 7. Clenching of the hands via controller grabs an object if it is close enough. The user will grab the object if the given hand's collider is intersecting the collider of a grabbable object. Currently, the application lacks tomato presence, which is the phenomenon where an object is accepted as a stand-in of hands[13]. While tomato presence would hide hands, making objects easier to see while being moved, it does not add any functional benefit to the tool.

Fig. 7. Grabbing Cube with Raycasting

The user also has the option to bring an object closer or farther away from him or herself by pointing a casted ray at a collidable object. The casted ray is a part of the XR Interaction Toolkit and more specifically the Interactor Line Visual script. Once the ray (which is at first red) hits a collidable GameObject, such as a relation cube, it will turn white to signify a selection. The user can

[13] https://owlchemylabs.com/tomatopresence.

then grab the object to move it at a distance. The object can move forward and backward in a range from the user's hand to far outside the visible room. While the ray is white, i.e., the user is holding the object via the ray, the held object can rotate on a single axis controlled by the player. Additionally, wherever the user points the ray the object will follow on its axis. The player can move both hands at the same time, which means that a user can pick up two objects and hold them at the same time.

5.4 Join Operation

An intuitive interaction in the tool can take place because the user can pick up and move objects. This interaction is the touching of two relation cubes in order to perform a JOIN operation. Given two relation cubes in a scene, the user can pick up both of them and move them close enough to where each of the cube's colliders is intersecting one another. A list is contained within each relation cube's attached script. If a collided object touches this relation cube it will be added to the relation cube's collision list. When the collision ends, i.e., the relation cubes' colliders no longer intersects, the collided object will be removed from the collision list. If another relation cube is determined to be touching this cube then both of the cubes turn green as seen in Fig. 8. The player can freely release the relation cubes, which will not be detached. In other words, the cubes will maintain the same location in Unity's 3D space. They also turn green.

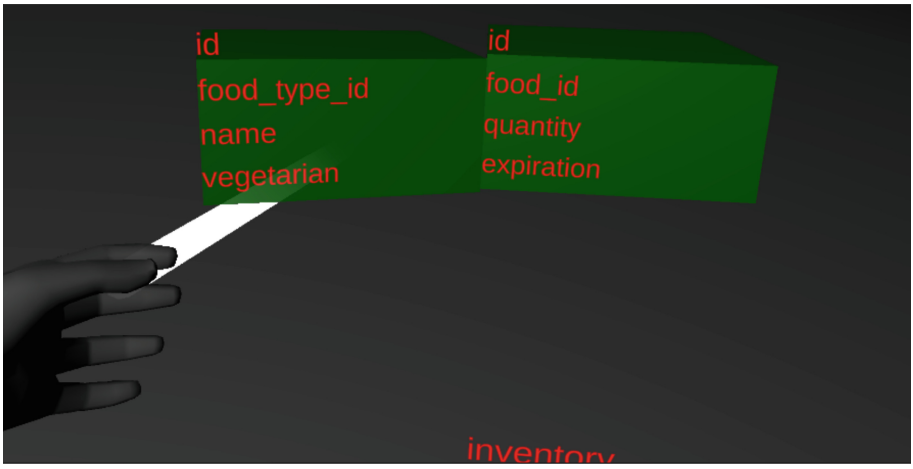

Fig. 8. JOIN Operation, each Relation Turns Green

Now the user can move his or her right hand near a green cube and a JOIN will occur. The resulting query output will display the joined data on the wall. Otherwise, if it does not have another relation cube (not green), then it will just select all data from that relation and display the data one of the wall. Listing 4

shows in line 1 that if there is a collision, then in lines 2–6, either a plain select or a join will occur (if cubes are touching - line 3). The JOIN result can be seen in Fig. 9.

```
1 collision(ob):
2   if obj is relation cube:
3     if obj.colliders includes relation cube:
4       print join query result to wall
5     else:
6       print select query result to wall
```

Listing 4. Query action collision

Fig. 9. JOIN Operation Result

6 Conclusions and Future Work

This paper described an approach to create and execute database queries by means of VR. This paper focused on describing the underlying technologies and technical aspects of the integration of Oculus Rift-S (VR headset and controllers), Unity (3D project platform), and SQLite (DBMS). An example demonstrates the feasibility of our approach with two simple query operations, select and join. Currently, this work does not allow for the selection of attributes of a relation; instead it queries for all fields.

As future work, an improvement would be to allow the user to select specific attributes to query for. This feature would also be helpful for the JOIN operation, as the current implementation performs a JOIN with no condition, effectively just a Cartesian product. Adding attribute selection would require designing more natural gestures for such operations. Moreover, new approaches will be explored to handle high-dimension problems, such as in the case of data cubes, which are multi-dimensional arrays of values. Also, experiments will be applied with users that interact with our VR approach to query large databases. A comparison will be carried out between the immersive VR approach and traditional queries on current 2D tools. Finally, the tool described only connects to a SQLite database. Future work could include using connectors to other DBMSs including MySQL, Microsoft SQL Server, PostgreSQL, and more.

References

1. Silva, J.N.A., Southworth, M., Raptis, C., Silva, J.: Emerging applications of virtual reality in cardiovascular medicine. JACC: Basic Transl. Sci. **3**(3), 420–430 (2018). publisher: American College of Cardiology Foundation. https://www.jacc.org/doi/abs/10.1016/j.jacbts.2017.11.009
2. Schleußinger, M.: Information retrieval interfaces in virtual reality-a scoping review focused on current generation technology. PLoS ONE 16(2), 1–24 (2021). https://doi.org/10.1371/journal.pone.0246398
3. Azzag, H., Picarougne, F., Guinot, C., Venturini, G.: VRMiner: a tool for multimedia database mining with virtual reality. In: Darmont, J., Boussaid, O. (eds.) Processing and Managing Complex Data for Decision Support, ch. 11, pp. 318–339. IGI Global (2006)
4. Oberhauser, R.: Database model visualization in virtual reality: exploring WebVR and native VR concepts. Int. J. Adv. Software **12**(3 & 4), 201–215 (2019)
5. Watanabe, C., Osugi, A., Masunaga, Y., Joe, K.: Queryball: a new model for querying in immersive VR systems. In: Arabnia, H.R., Mun, Y. (eds.) Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2003, 23–26 June 2003, Las Vegas, Nevada, USA, vol. 3, pp. 1106–1112. CSREA Press (2003)
6. Lochhead, I., Hedley, N.: Designing virtual spaces for immersive visual analytics. KN-J. Cartography Geographic Inf. **71**(4), 223–240 (2021)
7. Sassa, H., Cordeil, M., Yoshida, M., Itoh, T.: Visual linking of feature values in immersive graph visualization environment. In: The 14th International Symposium on Visual Information Communication and Interaction, Potsdam Germany, pp. 1–6. ACM, September 2021
8. Li, Y., Huang, J., Tian, F., Wang, H.-A., Dai, G.-Z.: Gesture interaction in virtual reality. Virtual Reality Intell. Hardware **1**(1), 84–112 (2019)
9. Chen, M., Peljhan, M., Sra, M.: EntangleVR: a visual programming interface for virtual reality interactive scene generation. In: Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology, pp. 1–6 (2021)
10. Zhang, C., Zeng, W., Liu, L.: UrbanVR: an immersive analytics system for context-aware urban design. Comput. Graphics **99**, 128–138 (2021)
11. Pereira, G.: Civic poverty and recognition. In: Schweiger, G. (ed.) Poverty, Inequality and the Critical Theory of Recognition. PP, vol. 3, pp. 83–106. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45795-2_4

12. Jin, Y., Ma, M., Zhu, Y.: A comparison of natural user interface and graphical user interface for narrative in HMD-based augmented reality. Multimedia Tools Appl. **81**, 5795–5826 (2021)
13. Lapointe, J.-F., Valdés, J.J., Belliveau, L., Vinson, N.G., Emond, B., Léger, S.: A distributed multimodal multi-user virtual environment for visualization and query of complex data. In: Ahram, T., Taiar, R., Gremeaux-Bader, V., Aminian, K. (eds.) IHIET 2020. AISC, vol. 1152, pp. 213–218. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44267-5_32