Storing and Querying Bitcoin Blockchain Using SQL Databases

Kwok-Bun Yue
yue@uhcl.edu

Karthika Chandrasekar
Chandrasekark@uhcl.edu

Hema Gullapalli
GullapalliH2010@uhcl.edu

Department of Computing Sciences
University of Houston-Clear Lake
Houston, TX 77058, U.S.A

Abstract

Bitcoin is the first major decentralized cryptocurrency with wide acceptance. A core technological innovation of Bitcoin is blockchain, a secure and pseudonymous general ledger that stores every Bitcoin transaction. Blockchain has received enormous attention from both the commercial and academic worlds, and it is generally recognized as the enabling technology of the Internet of Value (IoV), in which securely stored valuable entities are intended to be transferred as easily as information. Current blockchains are designed as special kinds of Online Transaction Processing (OLTP) systems, but not Online Analytical Processing (OLAP) systems. Data analytics by querying the blockchain directly can be ineffective. To incorporate the increasingly important blockchain technology into Information Systems curriculum, one approach is to store blockchain data in a SQL database, thus allowing fast data access and a simpler understanding of the underlying concepts. This paper describes our experiment of using three different methods for storing and querying Bitcoin data from SQL databases. It elaborates an assignment of querying a Bitcoin's SQL database in an undergraduate database course. The paper discusses our experience on using SQL databases for blockchain analysis, elaborates the characteristics of Bitcoin blockchain that make it an interesting database case, examines the relative merits of the three different methods, and provides suggestions on how they may be used in IS courses. Overall, we find that using SQL to query blockchains can be an effective educational technique for introducing it to IS curriculum.

Keywords: Blockchain, SQL, Bitcoin, database, query, data analytics.

1. INTRODUCTION

Bitcoin (Nakamoto, 2008) is the first major decentralized cryptocurrency with wide acceptance. It solves the double spending problem, in which a digital currency may be spent two or more times, by storing a publicly accessible general ledger of *all* Bitcoin transactions in a blockchain (Nakamoto, 2008).

Unlike bank transactions, Bitcoin transactions are digitally signed and irreversible, and are stored in a peer-to-peer network of nodes (running Bitcoin Core) using the Bitcoin protocol (Antonopoulos, 2017). Bitcoin Core (Bitcoin.org, 2018) is open sourced and contains code storing and maintaining a copy of the Bitcoin blockchain in a node, together with a reference Bitcoin's client to interact with the blockchain.

Although considered as behaving more like a speculative investment than a currency by many (Yermack, 2015; Detrixhe, 2018), Bitcoin has stormed into public awareness, reaching a historical peak price of $17,900 on December 15, 2017 (Wikipedia, 2018). Table 1 contains a collection of some vital Bitcoin parameters on 1/15/2019 11:00am central time to provide an illustrative snapshot. The data is collected from various public websites, including blockchain.info, bitnodes.earn.com, and bitcoinblockhalf.com. Some parameters will be explained later in Section 2. As of January 2019, the current size of the Bitcoin blockchain is more than 199GB and it stores all of the more than 373 million of Bitcoin transactions. With an all-time high market capitalization of $281 billion reached in December 2018 seemingly pulling out of thin air, no wonder Bitcoin has caught the imagination of the public.

Number of Bitcoin nodes	10,176
Number of Bitcoins mined	17,483,325 (83.25% of total)
Bitcoin's price	$3,632
Bitcoin's market capitalization	$64,292,208,028
Bitcoin blockchain's size	199.0GB
Latest block	558,665
Number of transactions in the latest block	2,854
Estimated transaction volume in the latest block	3,530.81722584 Bitcoin (BTC)
Total transaction fees in the latest block	0.14961742 BTC
Total number of all Bitcoin transactions	373.2 millions
Difficulty level	5,883,988,430,955
Number of transactions in the last 24 hours	321,592
Number of unspent transaction outputs	60,947,620

Table 1. A snapshot of Bitcoin's Parameters on 1/15/2019 11:00am central time

Bitcoin's success triggered many other cryptocurrencies, called altcoins, which numbered in 1,565 as of April 20, 2018 (Wikipedia 2018b). Even so, many consider that the blockchain technology developed and validated by Bitcoin may be much more important than Bitcoin itself (Tucker, 2018). Bitcoin blockchain can be considered as the first generation of blockchain that stores a specific cryptocurrency. Current and future generations of blockchains advance in many directions (Zheng, et al., 2017).

With the general ledgers of transactions nearly impossible to tamper with, blockchains can be extended to store any valuable property or asset beyond cryptocurrency. An important advancement is the introduction of rich programming languages and stateful blockchains to allow the constructions of smart software contracts to govern transaction completion, such as the approach taken by Ethereum (2018), the second most popular cryptocurrency. Another key development is permissioned blockchains, such as Hyperledger's fabric (Androulaki, et al., 2018). Unlike public blockchain such as Bitcoin in which everyone can participate, permissioned blockchains allow only a set of known and identified participants, who shared a common goals but do not fully trust each other. This enable private and federated blockchains (Zheng, et al., 2017).

Tapscott, & Tapscott (2017a) indicate that blockchain technology enables businesses with the Internet of Value (IoV): "a secure platform, ledger, or database where buyers and sellers could store and exchange value without the need for traditional intermediaries." The results can be drastically reduced transaction cost and friction that disrupts the usual ways of conducting businesses in a wide spectrum of areas. Using higher education as an example, blockchain allows a Web of decentralized transactions, possibly enabling huge changes in keeping student records, optimizing student loan management, improving pedagogy, incubating meta-universities, and ultimately creating a global network of learning institutes (Tapscott & Tapscott, 2017b). However, it is worthy to note that like many other leading edge technology, blockchains come with risks and costs (for example, see Walch, 2015).

Despite its importance, information systems (IS) research in blockchain is just beginning to emerge (Beck, Avital, Rossi & Thatcher, 2017). In IS education, blockchain can be relevant to many courses, including technical topics such as computer security, data analytics, databases, cryptocurrency, smart contracts, financial technology, etc. There are very few papers on incorporating blockchain technology in information systems and computing courses, especially in the lower level. An exception is (Delmolino, et al., 2016) that describes the experience of safe smart contract development laboratories in a security class. There is a gap between the importance of blockchain, and its existing body of knowledge and results in IS education. For example, the 2017 EDSIG

conference provided a workshop on "the Easy Way to Create a Blockchain using Fabric Composer" (Foley & Decker, 2017) in an effort to bridge the gap. This paper aims to contribute in filling this gap by describing our experience with storing and querying Bitcoin blockchain using SQL databases. It is possible that other popular blockchains, such as Ethereum, can be used for the same purpose of experimentation with blockchain. However, we selected Bitocin since it is the most popular public blockchain with tools widely available.

The rest of the paper is structured as follows. Section 2 discusses the basics of Bitcoin and its blockchain, and the goals of this work as the background context. Section 3 examines three methods of accessing Bitcoin data from SQL databases. Section 4 describes a Bitcoin's SQL query assignment in an undergraduate database course and the accompanying surveys. Section 5 discusses our experience using SQL to query the Bitcoin blockchain. It describes the characteristics that make it an interesting database case, and provides suggestions on how these different methods can be adopted in IS courses. Section 6 discusses future directions and draws our conclusions.

2. BACKGROUND

2.1 Bitcoin Blockchain and Transactions
Bitcoin blockchain stores the entire history of Bitcoin transactions. A transaction stores the transfers of Bitcoins (in the unit of Satoshi, with 1 Bitcoin (BTC) = 100,000,000 Satoshi) from input accounts to output accounts, plus authorization and other information. Bitcoin account addresses are public key hash values that can be authenticated by the corresponding private keys. Users can use a Bitcoin wallet to manage their Bitcoin accounts (public key hashes) and interact with the Bitcoin blockchain.

Unlike a bank transaction transferring money from one account to another account, Bitcoin transactions allow multiple inputs and multiple outputs. Figure 1 shows four historically interesting Bitcoin transactions. Figure 1a is the very first Bitcoin transaction as 50 BTC went to the Bitcoin address '1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa', which is assumed to be controlled by Satoshi Nakamoto, the mysterious Bitcoin's inventor(s). Bitcoin blockchain is known to be pseudonymous as all transactions are publicly accessible but the ownerships of accounts are anonymous within the blockchain. Many Websites provide Web pages

and APIs to access Bitcoin's data in various formats such as HTML, JSON or XML. For example, one can copy and paste Bitcoin addresses, transaction hash addresses, or block addresses from this paper into the popular site, Blockchain.info. Figure 2 shows a part of the output page of Blockchain.info for the Bitcoin transaction of Figure 1a.

Transactions are grouped in blocks. For example, Table 1 indicates that the block #558,665 has 2,854 transactions. The first Bitcoin transaction, called the *genesis transaction* here, is included in the first block (known as the Genesis Block or Block #0) as shown in Figure 1a. It is known as a *Coinbase transaction* to reward 50 BTC to the Bitcoin miner who had successfully created the block. Since the reward is created out of nowhere by Bitcoin, there is no input in a Coinbase transaction. Bitcoin mining involves finding a small enough block hash of the 80 Bytes header of the new block. The required smallness, or difficulty level, of the block hash is adjusted every 2,016 blocks to ensure that every block is mined in about 10 minutes. The difficulty level of 5,883,988,430,955 in Table 1 indicates a difficulty level of more than 5 trillion times as difficult as that of the Genesis block. The 80 Byte block header contains the hash of the Merkle tree which is constructed from the hashes (addresses) of all transactions, ensuring that transactions cannot be changed. The block header also contains the previous block hash and thus the block is *chained* together. Changing a block will change its block hash, and any subsequent block hashes will needed to be recomputed. This ensures that the blockchain is nearly impossible to tamper with.

Unlike a bank that keeps the balance of every account, Bitcoin blockchain keeps track of every transaction, including those transaction outputs (TXOut) that have not yet been spent, which are known as *unspent transaction outputs* (UTXO). UTXO can be used for future transaction inputs (TXIn). Note that in Figure 2, the transaction output of the Genesis block is still an UTXO. Thus, the very first Bitcoins generated has not yet been spent, probably intentionally.

Figure 1b shows another famous transaction, a1075db55d416d3ca199f55b6084e2115b9345e 16c5cf302fc80e9d5fbf5d48d, the first documented purchase of a good with Bitcoin in which 10,000 BTC was used to buy two Domino's pizzas on May 17, 2010. This *pizza transaction* has one TXOut (presumably going to an account owned by the pizza provider). Note that the buyer gathered together 131 UTXO from previous

transactions as TXIn to pool together 10,000.99 BTC. This paid the 10,000 BTC to the TXOut, and the transaction fee of 0.99BTC, which was collected by the block miner together with the 50 BTC mining reward. After the transaction was confirmed, these 131 UTXO were recorded as *spent* and can no longer be used as inputs to other transactions, thus solving the double spending problem. The current approximate 60,947,620 UTXO indicated in Table 1 is how the current Bitcoins are 'stored.'

Figure 1c shows how the 10,000 BTC were used by the 'pizza person' to provide for two TXOut in a transaction called the *pizza-provider transaction* here. Again, after this transaction, the previously unspent TXO to the pizza person with 10,000 BTC was recorded as spent.

In general, transactions can have multiple inputs and multiple outputs. Figure 1d shows the oldest Bitcoin transaction with three TXIn and two TXOut such that one TXOut address is also a TXIn address in the transaction (418b84d7649055411d8be4e241376a93825c1d 6248a304ae693060b3007a43f2). The sender gathered three UTXO in his accounts, each with 50 BTC. One TXOut received 105 BTC, and the *change* of 44.74 BTC, after 0.26 BTC transaction fee, was sent back to the address 1NA7Mopi9b4YhuWSBrB7D4W5XsTY53N1zY, which is one of the input addresses owned by the sender. We refer to this transaction as the *3i2o-change transaction*.

2.2 Purpose of Investigation
The technical details of Bitcoins are quite complicated and tedious. Much of the complexity of Bitcoins is owed to the complex decentralized and secure ledger structure, performance requirements, and constant evolutions of the Bitcoin software and protocol to solve emerging problems. In a sense, Bitcoin is a giant software experiment. Only few people, such as cryptocurrency developers and blockchain engineers, need to know many of these low-level and tedious complexity. For IS education, most students only need to know the basic blockchain structure, which can be modeled in a high level as containing a sequence of blocks of transactions, with each transaction having possibly multiple TXIn and TXOut, in which an UTXO from a previous transaction is used as the source for a TXIn (see Figure 3). Many IS courses may only need to use this high level model.

Blockchains make very good cases for data science and analytics courses. For example, one may search using the keywords 'blockchain' or 'bitcoin' in the leading data science and analytics site Kaggle (2018), and find vibrant communities with a large collection of datasets and kernels. However, current blockchains are designed mostly as special kinds of Online Transaction Processing (OLTP) systems, but not Online Analytical Processing (OLAP) systems. Data analytics by querying the blockchain directly, such as using the reference Bitcoin's client, can be ineffective (Anh, et al., 2018). Although many Websites provide services for querying Bitcoin blockchain, they are mostly limited by their usage policies and interfaces, and can be effective only for small queries that do not process a large numbers of transactions. Furthermore, the result formats may not be suitable for analytics.

Therefore, there are much activity on extracting data from Bitcoin for storage in databases that can provide efficient accessing. For example, McGinn, McIlwraith & Guo (2018) and Spagnuolo, Maggi & Zanero (2014) both used Neo4j, an open source graphical database. In this work, we select to use SQL databases to construct examples and assignments for accessing, querying, and analyzing Bitcoin. SQL is a high level declarative language that is relatively easy to learn. Students with some database background should be familiar with it. With highly available SQL developers, it has become a de facto standard even for many non-relational databases. For example, in Big Data technologies, HiveQL is a SQL-like declarative language of Hive for MapReduce (Thusoo, et al., 2010), and Spark-SQL is a SQL dialect on top of Spark (Armbrust, et al., 2015). Similarly, cloud computing platforms also embrace SQL, such as BigQuery by Google (2018a), which supports an extension of standard SQL. Thus, our purpose is to investigate using SQL databases in IS courses for querying blockchains.

3. ACCESSING BITCOIN DATA WITH SQL

This section describes three methods we have investigated: Abe-Bitcoin, BigQuery's Bitcoin, and blockchainsql.io (bcsql). We identified a collection of query problems for Bitcoin and developed solutions on these methods as a practical way to examine them for suitability of setting assignments. The near term goal was to identify a suitable platform for assignments in an undergraduate database course.

3.1 A Local Bitcoin SQL Database
Storing the Bitcoin blockchain in a *local* SQL database allows full control and customization to satisfy diverse needs. Bitcoin blockchain is an *append-only* database in which the only change

occurs about every ten minutes when a new block is created. Blocks are stored by Bitcoin Core in data files that do not change (except for the most recent evolving one) and can be parsed to populate a SQL database. There are available open source Bitcoin SQL database options, such as Abe (2018) and Bitcoin Database Generator (2018). We selected Abe because it captures more blockchain data, is more popular, and can also be used to store a number of other cryptocurrencies.

To install Abe, it is necessary to install Bitcoin Core to obtain a local copy of the blockchain first. Depending on the connection bandwidth and computer configuration, it may take a few hours to a few weeks to fully synchronize with the Bitcoin network. We selected Postgres 9.6 to install Abe because it has good performance properties and provides many modern SQL constructs. Abe is still in an Alpha version and we had to overcome a few technical issues. Eventually, the installation was complete but it took many days to do so in an old notebook.

Partial ER diagrams of the three methods are shown in Figure 4 in Appendix 1. Table 2 shows some of their basic parameters. Abe has 17 tables and 4 views. It is designed to be flexible enough to handle multiple cryptocurrencies. Many tables do not have derived columns that are computed and stored for efficiency. For example, the table txin(txin_id, tx_id, txin_pos, txout_id, txin_scriptsig, txin_sequence) stores information about transaction input. The field txin_id serves as a surrogate primary key, and tx_id and txout_id are foreign keys referencing the transaction containing the txid, and the txout used for the TXIn respectively. The other three columns are basic raw data. Users accessing a TXIn usually needs more than raw basic data and Abe uses a view txin_detail, which has 21 columns to provide contextual and summary data for the TXIn.

	Abe	BigQuery	bcsql
# tables	17	2	13
# views	4	0	0
# stored derived columns	5	0	15

Table 2 Some Parameters of the Three DB

To provide some ideas of how queries can be constructed, consider the following four problems, each related to an example transaction in Figure 1.

1. Genesis transaction: find the (Genesis) block hash from the transaction hash.

2. Pizza transaction: find the addresses and amounts of the TXIn from the pizza transaction hash.

3. Pizza-provider transaction: find the pizza-provider transaction hash, its output addresses, and amounts that used the UTXO of the pizza transaction.

4. 3i2o-change transaction: find the transaction hash of the first transaction with 3 TXIn and 2 TXOut, and also with a change going back to one of the TXIn addresses.

For reference, Appendix 2 lists the solutions to these problems using Abe. During our investigation, we found the relation schema of Abe to be relatively easy to use and we were able to construct solutions for a good collection of interesting query problems, some significantly more complicated than the four examples here. However, there was a performance issue in Abe that can be crucial when used concurrently by many students, especially since many of them are relative novices. For example, the Abe's solution for the pizza provider transaction in Appendix 2 selects from six table instances, two of which being of the table txout. It once took 143 ms to execute in an old notebook. If we replace one table instance of txout by the view txout_detail, which provides additional contextual and summary columns, the query only needs to select from three more table instances, making the query simpler. However, the execution time became 13 minutes. This is more serious in the 3i2o-change problem. The solution in Appendix 2 limits the solution space to the first 500 transactions with three TXIn and the first 500 transactions with two TXOut and hopes that the intersection of these two pools of transactions includes the result, which it does. Removing these limits make the query not able to complete in hours. Thus, students submitting non-optimized SQL queries can clog up the database. We are currently working on improving the performance of Abe. Before its performance becomes more acceptable, it is desirable to use other methods for setting the assignments.

3.2 Through Cloud Computing

Google's BigQuery is a cloud based enterprise data warehouse platform for real time data analysis using SQL that is compliant to the SQL 2011 standard and it has extensions for querying nested data (Google, 2018a). Customers are charged by the number of bytes of data processed (*scan cost*) and the first 1TB per month is free. Controlling costs by minimizing the volume of

data processed of the query is a key concern in cloud computing (Google, 2018b).

BigQuery's extensions to SQL allow columns to store records and structures. Structures can be expanded into tables by using the UNNEST function, which can then be used like derived tables in the JOIN and SELECT clauses. Thus, its public Bitcoin's dataset has only two tables: blocks and transactions, with internal structures stored in columns. For examples, the many TXIn and TXOut of a transaction are stored in the columns 'inputs' and 'outputs' of the transaction respectively.

BigQuery's Bitcoin is designed mainly for fast data analytics using a columnar storage and tree architecture (Sato, 2012). Bitcoin data is filtered and selectively stored in ways to facilitate analysis for various kinds of analytic problems. It is however not designed for exploring individual transaction. The complexity for the solutions of the four problems is also higher since it does not generate a surrogate key for TXOut to easily link TXIn to TXOut. Thus, we decided not to use BigQuery as the platform for our database assignments that usually contain mostly OLTP type questions with some simple data analysis problems.

3.3 Through a Third Party Web Interface

We next investigated blockchainsql.io (2018), which contains a Web interface to submit SQL statements to query its proprietary SQL Bitcoin database. Figure 5 shows a screenshot where users can submit queries or inspect the relation schema. It uses Microsoft SQL Server and has 13 tables, with many stored derived columns to improve performance. It is sufficiently fast for the large majority of the problems we prepared for the problems, and students reported no performance issues.

Thus, with no setup and maintenance cost, reasonable performance, and ease of use, our first pilot assignment used blockchainsql.io. However, it is worthy to point out its limitations. The instructor has no control of its availability, reliability, or interface, and can only use whatever data the provider selects to provide. For example, the latest available block it provided on January 2019 was #487,853 with a timestamp of "2017-09-06 16:23:23." Thus, about 16 months of the most recent blocks were not available. Moreover, the output is in HTML and limited by the provider to 10 rows per page. It cannot be used easily as input for further data analysis. Despite these limitations, we found that blockchainsql.io is ideal for lightweight small database assignments.

4. A BITCOIN'S SQL ASSIGNMENT

We experimented with an assignment on using SQL to query Bitcoin blockchain with blockchainsql.io in an undergraduate Introduction to Database course in Spring 2018. It is homework #8 of a total of 10 assignments in the course. There was an earlier traditional SQL assignment. We gave a one hour lecture to introduce cryptocurrency, Bitcoin, and blockchain, but did not discuss blockchainsql.io as students were expected to explore it themselves. Studying and understanding existing relation schema is a course objective. Because of space, Appendix 3 shows only the core part of the assignment without the introductory narratives on Bitcoin, blockchain, and submission requirements.

The objectives of the assignments are:
1. Execute SQL statements via a third party Web interface.
2. Study the relation schema of a new application: a Bitcoin SQL database.
3. Gain insight on blockchain and Bitcoin.
4. Gain some exposure on Microsoft SQL Server. (The course mainly used MySQL.)

The assignment contains six query questions ranging from easy to beginning intermediate. Screenshots of expected output are provided with explanations. Tips and suggestions are included for the more difficult questions mainly on the difference between MySQL and MS SQL. Students need to have a good understanding of the relational schema to answer the questions, especially the more difficult ones. As a reference, the suggested solutions are shown in Appendix 4.

Before the lecture, a pre-assignment survey was conducted with 25 respondents. It shows that two students have personally invested in Bitcoin and 9 students have friends or family members invested in Bitcoin. This is a relatively high participation comparing to the general public.

In a post-assignment survey, students were asked about their perception on various aspects of the assignments in a scale of 7 (1 signifying strong disagreement, 7 strong agreement, and 4 neutral). The result is summarized in Table 3. Because of the small size of the sample, these results should only be considered to be preliminary. No quantitative analysis has been conducted.

Statement	Average
1. The assignment is useful.	5.58
2. The assignment is interesting.	6.00
3. The assignment is practical.	5.52
4. The assignment helps me gain experience on SQL execution through a Web interface.	5.65
5. The assignment exposes me to study the relational schema of a new application.	5.81
6. The assignment helps me gain insight on Bitcoin and blockchain.	5.94
7. The assignment helps me to gain experience on MS SQL Server.	5.74
8. Overall, the assignment is effective.	5.55

Table 3. Post-Assignment Survey Results

The average responses range from 5.52 to 6.00, suggesting that the assignment is relatively effective in achieving its learning objectives. The best response is on Q2 Interestingness (6.00). This suggests that a timely assignment on a confusing yet trending technology may be appealing. The response on Q3, help gaining insight in Bitcoin and blockchain, is also high at 5.94. This suggests this kind of assignments may be useful not only in a database course, but also in courses directly targeting cryptocurrency and blockchain.

We also asked two identical questions in both the pre-assignment and post-assignment surveys on the student's interest in Bitcoin and their familiarity on its technical aspects. The average responses in a scale of 5 are shown in Table 4.

	Pre	Post
Do you find Bitcoin interesting?	3.36	3.94
Are you familiar with the technical aspects of Bitcoin?	2.24	3.1

Table 4. Pre and post assignment surveys

There are marked improvements on both indicators after the assignment. However, since there were two events, the one hour lecture and the assignment, we do not know the portion of contribution from the assignment. Even with nearly no prior technical knowledge on Bitcoin, students seem to be doing fine in the assignments. The average grade for the assignment is 91.2, within the range of the average grades of 87.6 to 96.0 among the ten homework assignments. Overall, the surveys can only be considered as a pilot study but it points to the potential of using SQL to query Bitcoin blockchain as an effective learning tool.

5. DISCUSSION

As the perceived enabling technology of IoV, the next frontier in the advance of the Internet, blockchain is important in any forward looking IS curriculum. We discuss how blockchains can be incorporated into database courses as well as other IS courses in this section.

5.1 Blockchain as Database Cases
Bitcoin is not only technologically interesting, but is also a very good general case study. It is hard to find another application with such a high market capitalization and *all* transactions publicly accessible. Blockchains also make very good case studies for databases. Traditional database applications provide four basic functions of persistent data: create, read, update, and delete (CRUD). Normalization theory in relational databases aims at minimizing unnecessary data redundancy to better maintain data consistency while writing to the database (Elmasri, & Navathe, 2010; Ricardo, 2015). However, normalization may create more relations, resulting in degraded performance. Thus, when appropriate, there may be a reverse, denormalization process to improve performance. As data consistency is crucial for traditional database applications, most database courses and textbooks understandably pay much more attention to normalization than denormalization. On the other hand, in the era of Big Data, data in many newer applications is never updated or even deleted. These kinds of increasingly popular append-only databases have a strong effect of how databases should be designed and optimized but are not well treated in database courses. The Bitcoin blockchain is an excellent illustrative case study for append-only databases.

Because of space, we only discuss one other example here. Derived columns are an inadequately discussed topic in database education. As popular database textbooks, Ricardo and Urban (2015), and Elmasri and Navathe (2010) both discuss derived columns under ER-modeling in a single paragraph and both provide the classical example that 'age' is a derived column computed from the date of birth (dob). They emphasize that derived columns should not actually be stored, but computed every time when the values are needed. If, for example, age is actually stored, the functional dependency dob -> age will make the relation not in the third normal form, an indication of poor table design. This is the approach taken by Abe in

which there are only five stored derived columns, all in the table Block. Instead, Abe uses views to provide derived columns (such as total inputs and total outputs in a transaction), which are computed every time. The alternate option is to actually physically store the derived columns to avoid repeated computations. The stored derived columns will need to be recomputed whenever there are changes. For write-intensive databases, that can degrade the performance significantly. Thus, most DBMS called derived columns as computed columns and they are not physically stored by default. However, an append-only databases such as Bitcoin blockchain do not have this problem as stored derived columns will never be recomputed. Blockchainsql.io have many stored derived columns and its performance is generally superior. In contrast, the views in Abe are practically too slow for many queries.

Overall, we find that Bitcoin blockchain is a very unique and good case for database education.

5.2 Uses of SQL Databases in IS Courses
The three methods have their own relative merits. Having a local database storing the blockchain is the most versatile but also requires the most work. It is also desirable to install Bitcoin Core to include the actual Bitcoin blockchain for advanced experimentations. This is especially suitable for courses on computer security, cryptocurrency, or blockchain as low level assignments can better be designed using the additional query capability beyond the blockchain. For example, we have constructed queries to identify blocks with potential security events in the blockchain (such as attempted denial of service attacks) or altcoin transactions that piggyback on the Bitcoin blockchain.

Cloud computing solutions such as BigQuery are especially suitable for courses focusing on data analytics, data science, Big Data, and cloud computing. Students will have the additional stress on constructing code that optimizes cloud computing cost, but they are standard considerations for those topics. The Kaggle (2018) website on BigQuery's Bitcoin is especially good for data analytics and data science courses as it contains an active community of rich resources and kernels.

Finally, a website such as blockchainsql.io is especially suitable as a lightweight platform for database and other introductory courses. It is just necessary for the instructor to allow plenty of time for the assignment as the site is provided by a third party in which availability is not guaranteed.

6. CONCLUSIONS

This work can be considered as a pilot study on incorporating blockchain into IS curriculum. Though limited, the initial result is encouraging. We are working on many directions to extend the project and will report the results in the future.

1. Incorporate blockchain materials and assignments into other courses, especially related to data analytics, data science, and computer security.
2. Develop assignments on using the three methods discussed.
3. Develop a more versatile and effective local platform to support blockchain and Bitcoin experiments and assignments. This includes performance refinement of Abe, extension to include other relevant datasets (such as Bitcoin's price history) and cryptocurrencies, and the uses of other database systems, especially Neo4j and MongoDB.
4. Develop tools to support our local platform.
5. Create our own blockchain applications for experimentation.
6. Conduct a quantitative study on the effectiveness of the local platform and assignments.

In summary, incorporating an important enabling technology such as blockchain into the curriculum will increase the relevance of forward looking IS programs and this paper is a contribution on this direction.

7. REFERENCES

Abe, block browser for Bitcoin and similar currencies, Retrived May 8, 2018 from https://github.com/bitcoin-abe/bitcoin-abe.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Muralidharan, S. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference* (p. 30). ACM.

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., ... & Zaharia, M. (2015, May). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 1383-1394).

Anh, D. T. T., Zhang, M., Ooi, B. C., & Chen, G. (2018). Untangling Blockchain: A Data

Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*.

Antonopoulos, A. M. (2017). Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media, Sebastopol, CA.

Beck, R., Avital, M., Rossi, M., & Thatcher, J. B. (2017). Blockchain Technology in Business and Information Systems Research. *Business & Information Systems Engineering*.

Bitcoin Database Generator (2018). Retrieved May 8th, 2018 from https://github.com/ladimolnar/BitcoinDatabaseGenerator.

Bitcoin.org (2018). Bitcoin Core. Retrieved May 5, 2018 from https://bitcoin.org/en/.

Blockchainsql.io (2018). Front Page, http://blockchainsql.io/. Retrieved May 3, 2018.

Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2016). Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *International Conference on Financial Cryptography and Data Security* (pp. 79-94). Springer, Berlin, Heidelberg.

Detrixhe, J. (2018). Robert Shiller wrote the book on bubbles. He says "the best example right now is bitcoin." Retrieved May 5, 2018 from https://qz.com/1067557/robert-shiller-wrote-the-book-on-bubbles-he-says-the-best-example-right-now-is-bitcoin/.

Elmasri, R., & Navathe. S. (2010). Fundamentals of database systems. Addison-Wesley Publishing Company.

Ethereum (2018). Ethereum Homestead Documentation, Retrieved May 6, 2018 from http://www.ethdocs.org/en/latest/.

Foley, J., & Decker, M. (2017)The Easy Way to Create a Blockchain using Fabric Composer, Workshop Presentation, *Proceedings of the 2017 EDSIG conference*.

Google, BigQuery's front page (2018a), https://cloud.google.com/bigquery/, Retrieved May 4, 2018.

Google, BigQuery Best Practices: Controlling Costs (2018b), https://cloud.google.com/bigquery/docs/best-practices-costs

Kaggle, Bitcoin blockchain, Retrieved May 11, 2018 from https://www.kaggle.com/bigquery/bitcoin-blockchain.

McGinn, D., McIlwraith, D., & Guo, Y. (2018). Toward Open Data Blockchain Analytics: A Bitcoin Perspective. *arXiv preprint arXiv:1802.07523*.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved May 5, 2018 from https://bitcoin.org/bitcoin.pdf.

Ricardo, C. M. (2015). Databases illuminated. Jones & Bartlett Publishers.

Sato, K. (2012). An inside look at google bigquery. White paper, Retrieved May 19, 2018 from https://cloud. google. com/files/BigQueryTechnicalWP.pdf.

Spagnuolo, M., Maggi, F., & Zanero, S. (2014). Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security* (pp. 457-468). Springer, Berlin, Heidelberg.

Tapscott, D., & Tapscott, A. (2017a). How blockchain will change organizations. *MIT Sloan Management Review*, 58(2), 10.

Tapscott, D., & Tapscott, A. (2017b). The BlockChain revolution and higher education. *Educause Review*, 52(2), 11-24.

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., ... & Murthy, R. (2010). Hive-a petabyte scale data warehouse using hadoop. In *IEEE 26th International Conference on Data Engineering (ICDE)*, 2010 (pp. 996-1005).

Tucker, I. (2018). Blockchain: so much bigger than bitcoin… The Guardian. Retrieved May 5, 2018 from https://www.theguardian.com/technology/2018/jan/28/blockchain-so-much-bigger-than-bitcoin.

Walch, A. (2015). The bitcoin blockchain as financial market infrastructure: A consideration of operational risk. *NYUJ Legis. & Pub. Pol'y*, *18*, 837.

Wikipedia (2018a). History of Bitcoin. Retrieved May 5, 2018 from https://en.wikipedia.org/wiki/History_of_bitcoin.

Wikipedia (2018b). List of cryptocurrencies. Retrieved May 6, 2018 from https://en.wikipedia.org/wiki/List_of_cryptocurrencies.

Yermack, D. (2015). Is Bitcoin a real currency? An economic appraisal. In *Handbook of digital currency* (pp. 31-43).

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. *In Big Data (BigData Congress)*, 2017 IEEE International Congress on (pp. 557-564).

Editor's Note:

This paper was selected for inclusion in the journal as an EDSIGCON 2018 Distinguished Paper. The acceptance rate is typically 7% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2018.

Appendices

Appendix 1. Figures

Transaction   TXOut

Trans1   Addr1: 50 BTC

Trans1: 4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
Addr1: 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa (TXOut)

(a) The Genesis Transaction

TXIn   Transaction   TXOut

Addr1: 150 BTC   Trans1   Addr2:
                                                               10,000 BTC
....
Addr1: 400 BTC

---·--- 131 TXIn from Addr1 totaling 10,000.99 BTC

Trans1: a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d
Addr1: 1XPTgDRhN8RFnzniWCddobD9iKZatrvH4 (TXIn)
Addr2: 17SkEw2md5avVNyYgj6RiXuQKNwkXaxFyQ (TXOut)
Transaction fee: 0.99BTC

(b) The Pizza Transaction

TXIn   Transaction   TXOut

Addr1: 150 BTC   Trans1   Addr2:
                                                               5,777 BTC

                                                               Addr3:
                                                               4,223 BTC

Trans1: cca7507897abc89628f450e8b1e0c6fca4ec3f7b34cccf55f3f531c659ff4d79
Addr1: 17SkEw2md5avVNyYgj6RiXuQKNwkXaxFyQ (TXIn)
Addr2: 1MLh2UVHgonJY4ZtsakoXtkcXDJ2EPU6RY (TXOut)
Addr3: 13TETb2WMr58mexBaNq1jmXV1J7Abk2tE2 (TXOut)

(c) The Pizza-Provider Transaction

TXIn   Same address   TXOut

Addr1: 50 BTC   Addr1:
                              Transaction   44.74 BTC
Addr2: 50 BTC   Trans1

Addr3: 50 BTC   Addr4:
                                                               105 BTC

Trans1: 418b84d7649055411d8be4e241376a93825c1d6248a304ae693060b3007a43f2
Addr1: 1NA7Mopi9b4YhuWSBrB7D4W5XsTY53N1zY (TXIn and TXOut)
Addr2: 1J27CLhDGmm3qBSiVcGxoE3evoSECUREYj (TXIn)
Addr3: 1DJ8d8gVU5VFGpSjr2AzwS9Jtg5YnyfWQD (TXIn)
Addr4: 1P3CKNyDEMRKHTDTLPqesYKSzPCo1QUCQK (TXOut)
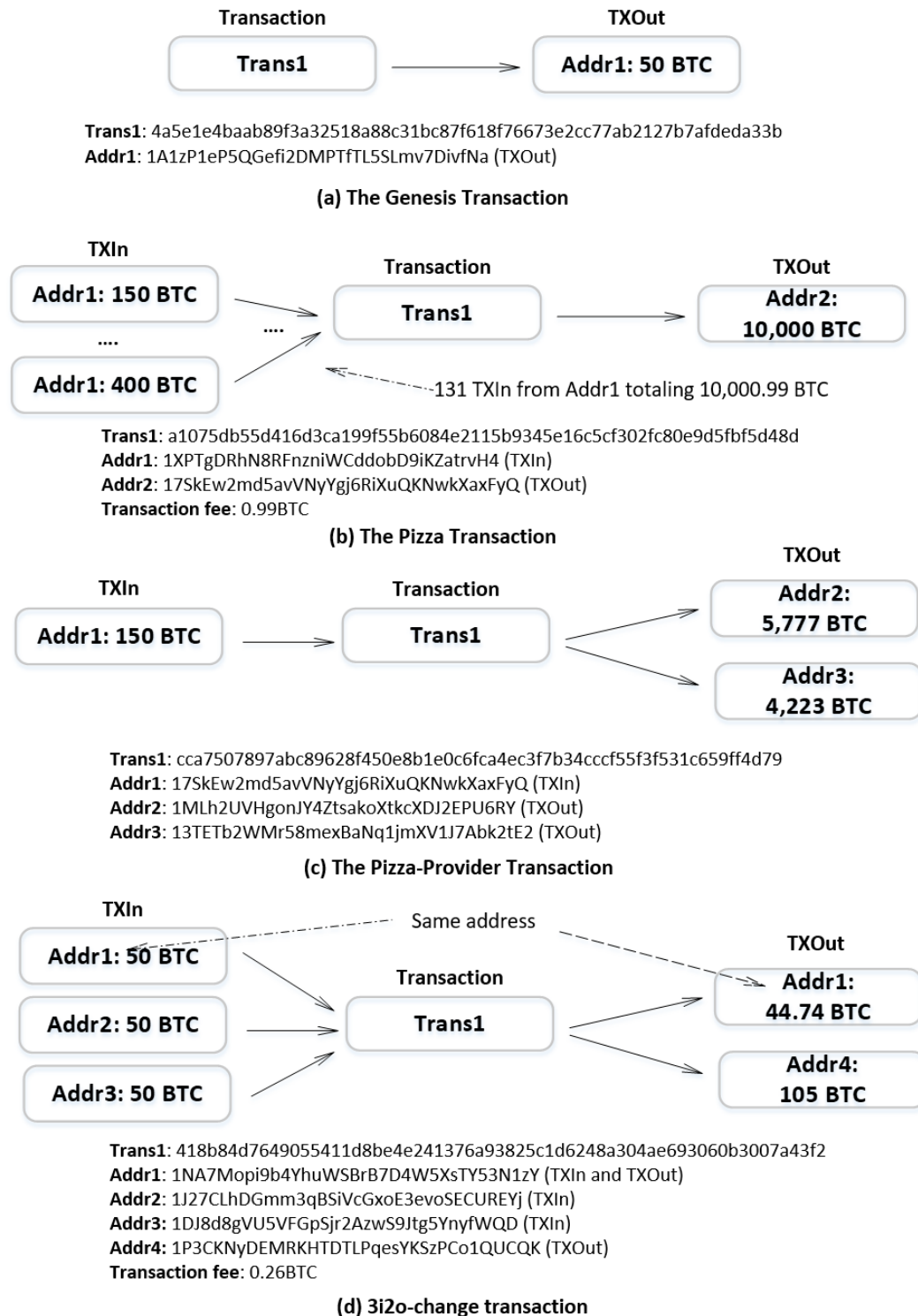**Transaction fee**: 0.26BTC

(d) 3i2o-change transaction

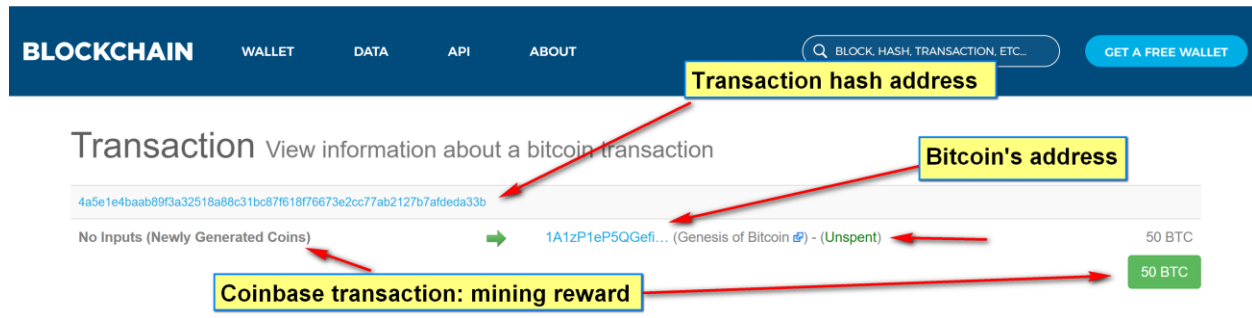**Figure 1. Four Interesting Bitcoin Transactions**

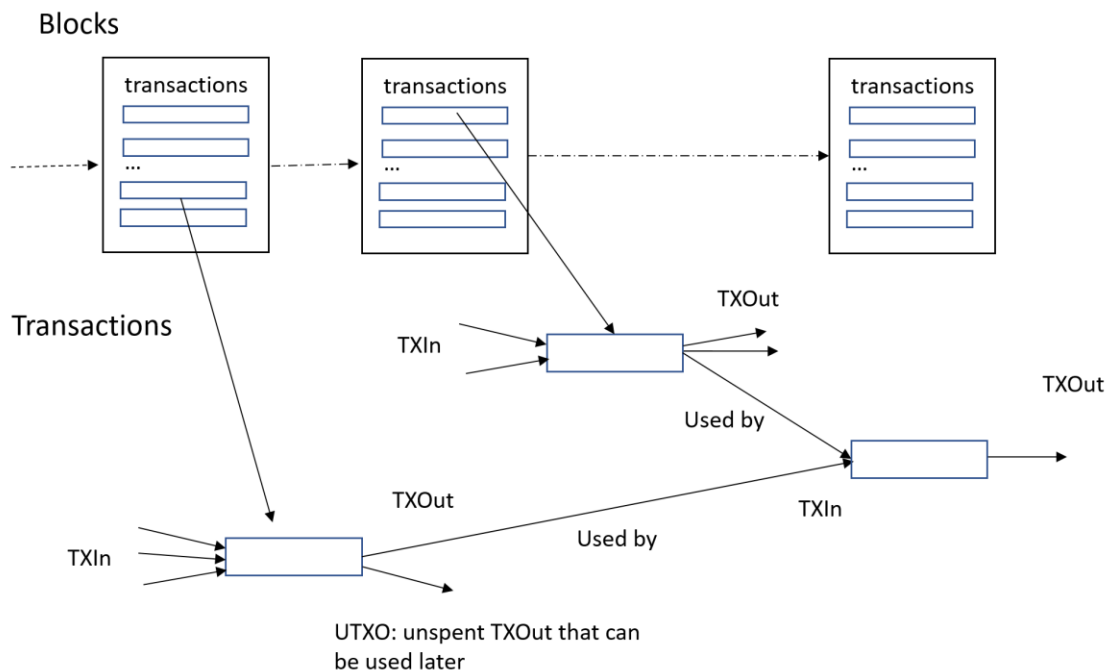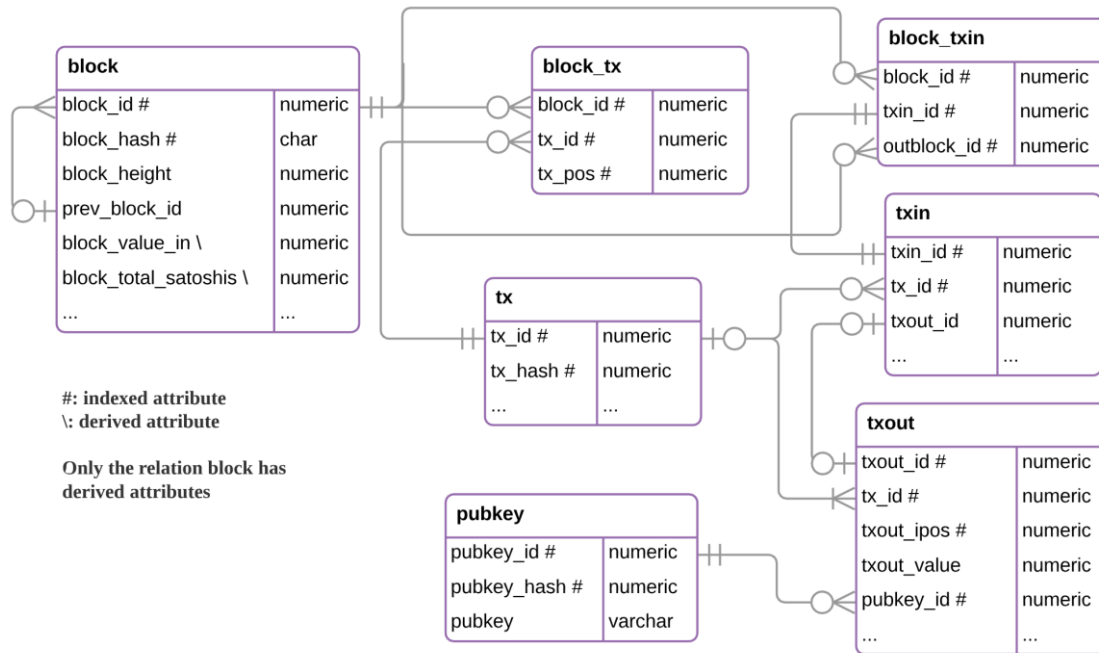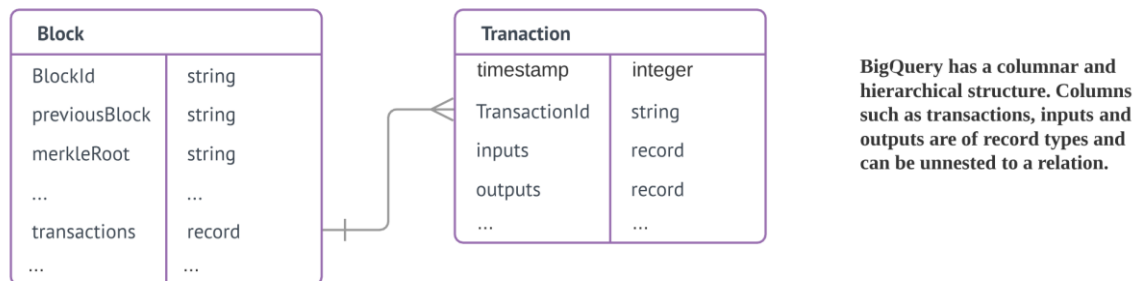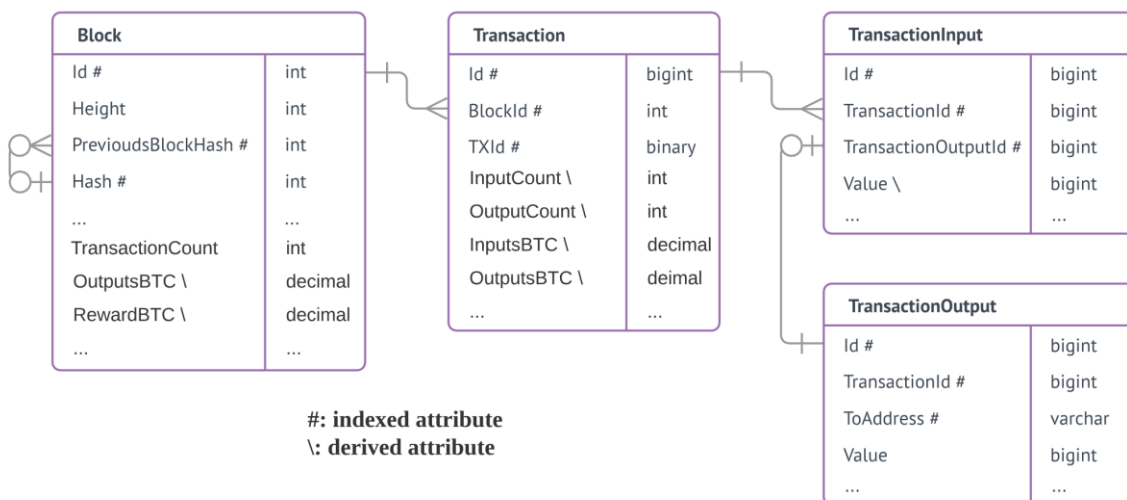**Figure 2. Information about the First Bitcoin Transaction Shown in blockchain.info**

Figure 3. A High Level Model for Bitcoin Blockchain

block

block_id #	numeric
block_hash #	char
block_height	numeric
prev_block_id	numeric
block_value_in \	numeric
block_total_satoshis \	numeric
...	...

block_tx

block_id #	numeric
tx_id #	numeric
tx_pos #	numeric

block_txin

block_id #	numeric
txin_id #	numeric
outblock_id #	numeric

txin

txin_id #	numeric
tx_id #	numeric
txout_id	numeric
...	...

tx

tx_id #	numeric
tx_hash #	numeric
...	...

txout

txout_id #	numeric
tx_id #	numeric
txout_ipos #	numeric
txout_value	numeric
pubkey_id #	numeric
...	...

#: indexed attribute
\: derived attribute

Only the relation block has derived attributes

pubkey

pubkey_id #	numeric
pubkey_hash #	numeric
pubkey	varchar

(a) Abe-Bitcoin

Block

BlockId	string
previousBlock	string
merkleRoot	string
...	...
transactions	record
...	...

Tranaction

timestamp	integer
TransactionId	string
inputs	record
outputs	record
...	...

BigQuery has a columnar and hierarchical structure. Columns such as transactions, inputs and outputs are of record types and can be unnested to a relation.

(b) BigQuery's Bitcoin

Block

Id #	int
Height	int
PrevioudsBlockHash #	int
Hash #	int
...	...
TransactionCount	int
OutputsBTC \	decimal
RewardBTC \	decimal
...	...

Transaction

Id #	bigint
BlockId #	int
TXId #	binary
InputCount \	int
OutputCount \	int
InputsBTC \	decimal
OutputsBTC \	deimal
...	...

TransactionInput

Id #	bigint
TransactionId #	bigint
TransactionOutputId #	bigint
Value \	bigint
...	...

TransactionOutput

Id #	bigint
TransactionId #	bigint
ToAddress #	varchar
Value	bigint
...	...

#: indexed attribute
\: derived attribute

(c) blockchainsql.io

Figure 4. Partial ER Diagrams of Abe, BigQuery-Bitcoin and Blockchainsql.io
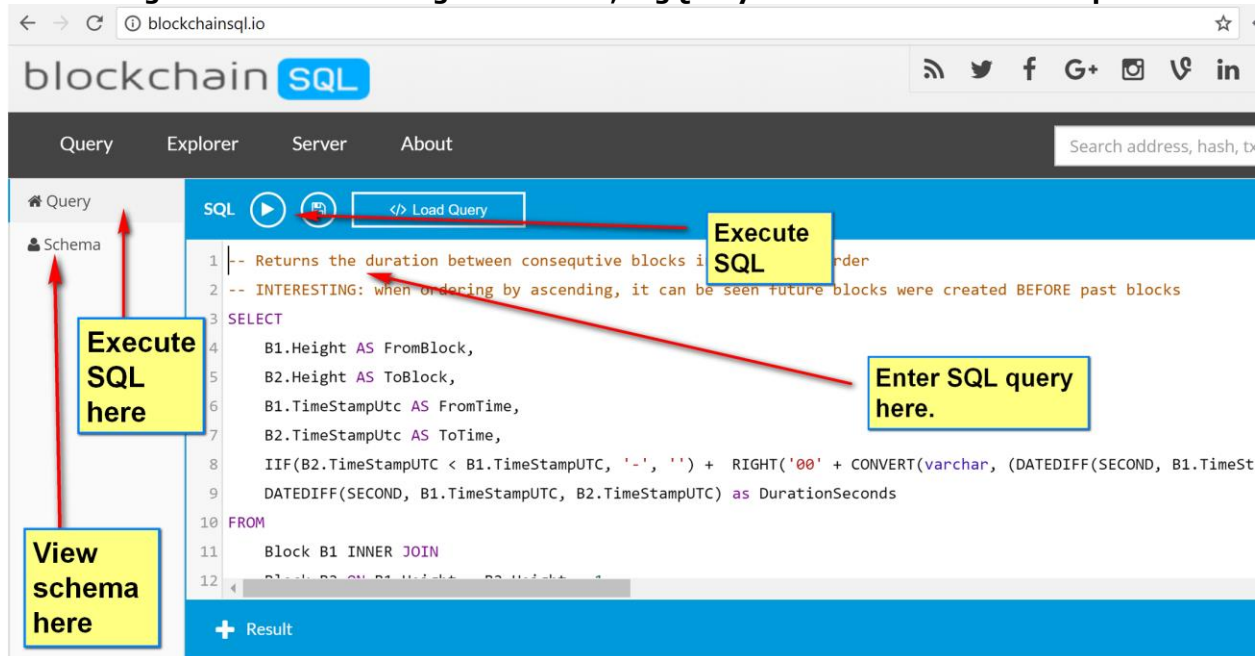

Figure 5. A Screenshot of the Front Page of Blockchainsql.io

Appendix 2 Abe's Solutions to the Four Query Problems.

1. Genesis transaction: find the (Genesis) block hash from the transaction hash.

```
select b.block_hash
from block b join block_tx bt on (b.block_id = bt.block_id)
    join tx t on (bt.tx_id = t.tx_id)
where t.tx_hash = '4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b';
```

2. Pizza transaction: find the addresses and amounts of the TXIn from the pizza transaction hash.

```
select i.txin_pos,  o.pubkey_hash, o.txout_value
from tx t join txin i on (t.tx_id = i.tx_id)
        join txout_detail o on (o.txout_id = i.txout_id)
where t.tx_hash = 'a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d'
order by 1 asc;
```

3. Pizza-provider transaction: find the pizza-provider transaction hash, its output addresses, and amounts that used the UTXO of the pizza transaction.

```
select t2.tx_hash as tx_hash,
    p.pubkey_hash as address,
    o2.txout_pos as position,
    o2.txout_value as amount
from tx t1, txout o1, txin i, tx t2, txout o2, pubkey p
where t1.tx_id = o1.tx_id
and i.txout_id = o1.txout_id
and o2.tx_id = i.tx_id
and i.tx_id = t2.tx_id
and o2.pubkey_id = p.pubkey_id
and t1.tx_hash = 'a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d'
```

4. 3i2o-change transaction: find the transaction hash of the first transaction with 3 TXIn and 2 TXOut, and also with a change going back to one of the TXIn addresses.

```
select distinct t.tx_id, t.tx_hash  -- depend on the fact that earlier transactions have smaller tx_id
from
  ((select tx_id from txin   -- first 500 tx with 3 inputs
    group by tx_id
    having count(txin_id) = 3
    order by tx_id
    limit 500)
    intersect
    (select tx_id from txout  -- first 500 tx with 2 outputs
    group by tx_id
    having count(txout_id) = 2
    order by tx_id
    limit 500))
    as temp  -- tx with 3 inputs and 2 outputs
join tx t on (temp.tx_id = t.tx_id)
join txin i on (temp.tx_id = i.tx_id)
join txout o1 on (i.txout_id = o1.txout_id)
join txout o2 on (temp.tx_id = o2.tx_id)
where o1.pubkey_id = o2.pubkey_id   -- output address is the change going back to an input address
order by t.tx_id
limit 1;
```

Appendix 3 A SQL Assignment on Querying the Bitcoin Blockchain

[1] Show the block information of the block with the hash address of
0x0000000000000000009769B8206EB613FBC90C607544636886E11CEB9161E33F.

Note that the hash address is the identifier of a block. Mining a Bitcoin block is more or less finding an acceptable block hash address with enough number of 0's at the beginning.

| Result | Executed On 2018-03-27 16:41:19 UTC   Duration 00:00:00.2867756 |

ID	Height	PreviousBlockHash	Hash	E
2856743	483857	00000000000000000087C230D5CDE4C326653EC5158A1AD1F24BFDC53B7CD7F	0000000000000000009769B8206EB613FBC90C607544636886E11CEB9161E33F	1

[2]   Show the height of the most recent block stored in http://blockchainsql.io/.

| Result | Executed On 2018-03-27 16:48:50 UTC   Duration 00:00:00.0560538 |

max_height
483857

[3]   Show the most recent block stored in http://blockchainsql.io/.

| Result | Executed On 2018-03-27 16:51:22 UTC   Duration 00:00:00.2407139 |

ID	Height	PreviousBlockHash	Hash	E
2856743	483857	00000000000000000087C230D5CDE4C326653EC5158A1AD1F24BFDC53B7CD7F	0000000000000000009769B8206EB613FBC90C607544636886E11CEB9161E33F	1

[4] Average number of transactions per block in the entire Bitcoin blockchain.

avg_transaction_count
521

[5] Average number of transactions per block for every year since the blockchain was created.

year	avg_transaction_count
2009	1
2010	2
2011	31
2012	155
2013	309
2014	429
2015	840
2016	1506
2017	1814

Tips: you may find the data type of the relevant column in the relation Block by following using the 'Schema' tab in the website.

[6] Show a summary report of the transactions in the block with height 400000 with three columns:

1. "# inputs in the group": number of inputs in each group of the result. A group result is shown in one row.
2. "Number of transactions": numbers of transactions with this number of inputs.
3. "Total input Bitcoins": total inputs' BTC of transactions with this number of inputs.

# inputs in the group	Number of transactions	Total input Bitcoins
1	794	23065.99413116
2	391	900.08223074
3	185	407.30818001
4	72	120.03196718
5		
6		
10	22	55.54584783
9	14	90.2052241

In descending order of

There are 22 transactions iwth 10 TXIn.
Their total input BTC is 55.54584783 BTC.

Tips:

1. In SQL Server, transaction is a keyword and cannot be used directly as a table name. You will need to refer to the transaction table as [transaction].
2. MS SQL Server is stricter than MySQL in many areas. For example,

select s.city, s.sname, count(s.snum)
from supplier s
group by s.city
order by count(s.snum);

is acceptable by MySQL without any syntax error. In MS SQL Server, there are a few syntax errors:

1. s.sname is not acceptable as a select column since it is not a group by column. MySQL may just output the first s.sname in the group (which is not semantically correct as there may be many supplier names for the same supplier city). MS SQL Server produces an error.
2. In MS SQL, a column needs to have a name, and therefore you will need to use "count(s.snum) as count".
3. As a group function, count(s.snum) cannot be used in the order by clause in MS SQL.

You will need to have your SQL statement in MS SQL as:

select s.city, count(s.snum) as count
from supplier s
group by s.city
order by count;

Note that count now refers to count(s.snum) and s.sname must be removed.

Appendix 4 Suggested Solutions for the Assignment (in MS SQL of blockchainsql.io)

[1]

```
select *
from Block
where hash = 0x000000000000000009769B8206EB613FBC90C607544636886E11CEB9161E33F
```

[2]

```
select max(height) as max_height
from Block
```

[3]

```
select *
from Block
where Height =
        (select max(height) as max_height
        from Block)
```

[4]

```
select avg(TransactionCount) as avg_transaction_count
from Block
```

[5]

```
select year(b.TimeStampUtc) as year,
        avg(b.TransactionCount) as avg_transaction_count
from Block b
group by year(b.TimeStampUtc)
```

[6]

```
select t.InputCount as "# inputs in the group",
        count(t.InputsBTC) as "Number of transactions",
        sum(t.InputsBTC) as "Total input Bitcoins"
from block b, [Transaction] t
where b.Id = t.blockId
and b.height = 400000
group by t.InputCount
order by "Number of transactions" desc
```