OligoArchive: Using DNA in the DBMS storage hierarchy

Raja Appuswamy, Kevin Lebrigand, Pascal Barbry, Marc Antonini, Oliver Madderson, Paul Freemont, James Macdonald, Thomas Heinis

OligoArchive: Using DNA in the DBMS storage hierarchy

Raja Appuswamy[1] Kevin Lebrigand[2] Pascal Barbry[2] Marc Antonini[3]
Oliver Madderson[4] Paul Freemont[4] James MacDonald[4] Thomas Heinis[5]

[1]Data Science Department, Eurecom, Biot, France
[2]Université Côte d'Azur & CNRS, Institut de Pharmacologie Moléculaire et Cellulaire
[3]Université Côte d'Azur & I3S, France
[4]Department of Medicine, Imperial College London, London, UK
[5]Department of Computing, Imperial College London, London, UK

ABSTRACT

The demand for data-driven decision making coupled with need to retain data to meet regulatory compliance requirements has resulted in a rapid increase in the amount of archival data stored by enterprises. As data generation rate far outpaces the rate of improvement in storage density of media like HDD and tape, researchers have started investigating new architectures and media types that can store such "cold", infrequently accessed data at very low cost.

Synthetic DNA is one such storage media that has received some attention recently due to its high density and durability. In this paper, we investigate the problem of integrating DNA in the database storage hierarchy. More specifically, we ask the following two questions: (i) how can database knowledge help optimize DNA encoding and decoding? and (ii) how can biochemical mechanisms used for DNA manipulation be used to perform in-vitro, near-data SQL query processing?

In answering these questions, we present *OligoArchive*, an architecture for using DNA-based storage system as the archival tier of a relational database. We demonstrate that OligoArchive can be realized in practice by building archiving and recovery tools (pg_oligo_dump and pg_oligo_restore) for PostgreSQL that perform schema-aware encoding and decoding of relational data on DNA, and using these tools to archive a 12KB TPC-H database to DNA, perform in-vitro computation, and restore it back again.

1. INTRODUCTION

Driven by the promise of data analytics, enterprises have started gathering vast amounts of data from diverse data sources. However, recent studies have pointed out that nearly 80% of data is "cold", or infrequently accessed, and is growing 60% annually, making it an ideal candidate for archival using cheaper storage devices [16]. Unfortunately, this rapid rate of data growth overshadows density improvement in traditional storage hardware like HDDs and tape. Thus, researchers have started exploring the use of hardware–software co-design techniques using existing storage media [3, 7, 14], or the use of new storage media that has radically different density and durability characteristics compared to HDD and tape [21, 28].

One such storage media that has received attention recently is DNA [6, 10, 12, 29]. DNA possesses four key properties that make it relevant for archival storage. First, it is an extremely dense three-dimensional storage medium that has the theoretical ability to store 455 Exabytes in 1 gram; in contrast, a 3.5" HDD can store 10TB and weighs 600 grams today. Second, DNA can last several centuries when stored at standard temperature in an anoxic, anhydrous atmosphere [13, 15]; HDD and tape have life times of five and thirty years. Third, it is very easy, quick, and cheap to perform in-vitro replication of DNA; tape and HDD have bandwidth limitations that result in hours or days for copying large EB-sized archives. The fourth property is the difference in Kryder's rate, or the rate at which media density improves every year. Kryder's rate is around 10% and 30% for HDD and tape [2, 11, 25]. Thus, if one stores 1PB in 100 tape drives today, within five years, it would be possible to store the same data in just 25 drives. As storage space is a premium commodity in datacenters, using tape for archival storage implies constant data migration with each new generation of tape. A recent article summarized the financial impact of such media obsolesence on the movie industry [22]. DNA does not have this problem as the Kryder's rate for DNA is theoretically zero given that the density of DNA is biologically fixed.

In this paper, we investigate the implications of integrating DNA in the DBMS storage hierarchy. We present *OligoArchive*, an architecture for using a DNA-based storage system as the archival tier of a relational DBMS. In doing so, we show how database engines and DNA storage can symbiotically work together to minimize storage cost and improve processing efficiency. In particular, we show how (i) database schema awareness can be used to improve density during the encoding process, and assist in identifying errors during the decoding process, (ii) SQL operations that are traditionally executed by the DBMS in-silico can be translated into biochemical techniques that can be executed in-vitro, thus enabling near-data query execution directly over DNA. We demonstrate the feasibility of OligoArchive by building tools for PostgreSQL (pg_oligo_dump and pg_oligo_restore) that can archive and restore relational data to and from DNA. Using these tools, we perform wet lab experiments in which we successfully archive a 12KB TPC-H data from a PostgreSQL database to synthetic DNA, perform in-vitro computation, and restore the data back again.

2. BACKGROUND

DNA, or Deoxyribo Nucleic Acid, is a macro-molecule that carries the genetic code required for functioning of all living organism. *Nucleotides* are the smaller molecules that form the building blocks of DNA. There are four types of nucleotides: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). Naturally occurring DNA is structured in the form of a double helix with two strands of nucleotides. In contrast, DNA used for data storage is a single stranded sequence of nucleotides, also referred to as an *oligonucleotide* (oligo), that is synthesized using a chemical process that assembles the DNA one nucleotide at a time.

DNA has directionality, and the two ends of a strand are referred to as the 5' and 3' ends. The nucleotide sequence encoded in the DNA is read out by *sequencing* the oligos. The first step involved in sequencing is amplifying the concentration of DNA via *Polymerase Chain Reaction* (PCR [4]). The PCR process takes as input

a template DNA strand, a polymerase enzyme, sequencing primers, and individual nucleotides. Sequencing primers are short strands of DNA used to indicate the exact beginning and end regions of the template DNA strand that must be replicated. Based on these guiding sites, the polymerase reads the nucleotide sequence from original single-stranded DNA and assembles the individual nucleotides to produce a complementary, parallel strand.

Following PCR, a DNA sequencer is used to read out the nucleotides. While there are several sequencing techniques, the most popular method is referred to as "sequencing by synthesis". At a very high level, this process also uses the DNA polymerase to synthesize a double stranded DNA from the single-stranded template similar to PCR. However, the nucleotides used during sequencing have special terminators that force the polymerase to assemble nucleotides one at a time. The assembled nucleotides are also fluoresced such that each nucleotide emits a different color. Thus, as the polymerase assembles nucleotides, fluorescent microscopy is used to record the emitted color. A base calling software uses such color readings to identify each nucleotide. Modern sequencing techniques can sequence a strand of DNA from both 5' and 3' ends. Such an approach, referred to as paired-end sequencing, is often preferred to single-end sequencing due to the ability to recover from sequencing errors due to higher coverage (the same oligo is read multiple times).

In this paper, we investigate using DNA as an archival storage media for databases based on the assumption that DNA storage will become cost effective. Such an assumption is not valid today as DNA synthesis costs approximately one-tenth to one-hundredth of a cent per nucleotide. Given that a nucleotide can essentially be treated as a quaternary code with four digits, the maximum information capacity of a nucleotide is 2 bits. Using $11/TB as an approximation for the cost of tape circa 2018 [2], DNA storage is at least seven orders of magnitude more expensive than tape. However, recent advances in sequencing and synthesis technologies portend significant improvements in DNA storage. It is well known that sequencing productivity has already overtaken Moore's law [8]. Promising work on new enzymatic synthesis technologies and parallel array synthesis techniques are expected to also make large-scale synthesis of oligos affordable in the future [17, 18].

3. OLIGOARCHIVE ARCHITECTURE

Database engines use a three-tier storage hierarchy that consists of devices with widely varying price/performance characteristics. The *performance* tier stores data accessed by high-performance OLTP and real-time analytics applications. Due to the stringent latency requirements of these applications, DRAM and Flash Solid State Storage Devices (SSD) are the typical storage media used in this tier. The *capacity* tier stores data accessed by latency-insensitive batch analytics applications. Hard Disk Drives (HDD) are typically the storage media of choice for this tier due to their low cost/GB compared to SSDs. Finally, the *archival* tier is used to store data that is accessed very rarely, for example, during security compliance checks, or legal audits. Virtual Tape Libraries (VTL) are the typical storage media used in this tier due to low cost and higher durability compared to HDDs.

The OligoArchive architecture alters the DBMS storage hierarchy by replacing the tape-based archival tier with a DNA-based one. In this section, we provide a high-level overview of the division of labor between the database engine and the DNA storage device, and the interface that a DNA storage device should expose to be used in OligoArchive.

3.1 Database Engine

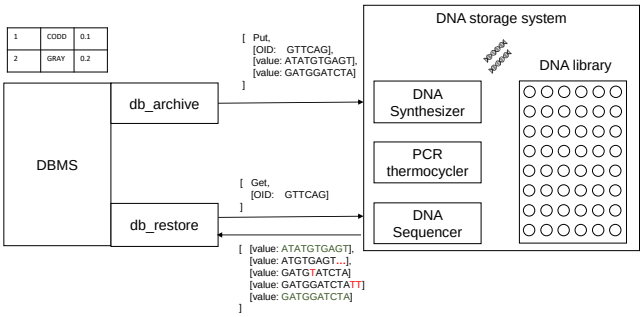

Figure 1: Database–DNA storage division of labor: The database system performs conversion between relational data and textual representation of oligonucleotide sequences. During a *put* operation, the DNA storage system synthesizes DNA strands and stores them in a library. During a *get* operation, DNA strands are sequenced and the reads are returned back. Both synthesis and sequencing are error prone as shown by truncated, extended, and erroneous reads during a *get*. The database encoding and decoding procedure should guarantee recovery despite such errors.

Database engines already have well establish procedures and tools for archiving data. PostgreSQL, for instance, provides tools pg_dump and pg_restore that can be used to selectively archive data from a database installation in a variety of formats. The database engine in OligoArchive is responsible for providing similar tools for converting data back and forth between the relational on-disk data format and oligonucleotide sequences. In order to do so, these tools should take into account several limitations of DNA synthesis and sequencing during encoding and decoding of data.

Data layout and random access. Due to limitations of DNA synthesis, the maximum length of each oligo ranges from a few hundred to a few thousand nucleotides at best. As the amount of data that can stored per oligo is in the order of a few Kilobits, a single archived table will be stored using thousands of oligos. Further, oligos themselves have no logical addressing like block-based disk and tape. Thus, addressing information must be encoded together with the data. Oligos also do not support random access ability, as it is not possible to resequence just a single oligo. Instead, random access is achieved by using a library of sequencing primers [6, 20, 29]. With such an approach, a group of oligos that form a logical unit (for instance, a record, a table, or a database) are associated with a single sequencing primer from the library of primers. The encoding procedure that converts relational data into nucleotide sequences should add the corresponding primers to each oligo. The right primer is then used during sequencing to selectively retrieve a portion of data, thus, providing random access.

Synthesis & sequencing errors. Oligos are synthesized chemically one nucleotide at a time. The probability that a nucleotide binds to the oligo during the synthesis procedure is called *coupling efficiency*. The coupling efficiency of synthesis techniques is related to the length of the oligos, with longer oligos suffering from more errors due to low efficiency. Synthesis also produces many truncated byproducts that are oligos that do not match the synthesis input. During sequencing, these spurious oligos might also be covered generating invalid data.

The sequencing procedure also introduces errors. The type of error introduced varies depending on the sequencing technology used. Short-read sequencers introducing more substitution errors, where the sequencer reports a wrong nucleotide, and long-read sequencers introduce indels, where the sequencer inserts or deletes spurious nucleotides. While modern sequencing technologies can

provide very high coverage, or error-free reads of oligos, it is well known that not all oligos get uniform coverage. Some oligos can be read thousands of times while others may not be sequenced at all. Thus, the encoder must add error correction metadata that can be used by the decoder to recover data back despite such errors, spurious additions and omissions.

Structural complexity. Oligos that contain homopolymer repeats (multiple consecutive occurrence of the same nucleotide), or high GC content (higher Gs and Cs than As and Ts) are known to amplify both synthesis and sequencing errors. Similar, current sequencers are also known to be prone to errors if all oligos in a pool contain low-complexity regions (same sequence of simple amino acids) at similar positions. Thus, the encoder should avoid generating such problematic oligonucleotide sequences.

3.2 DNA Storage System

The DNA storage system plays the role of traditional tape in OligoArchive in that it is responsible for storage of oligos. We model the DNA storage device as an object store. Similar to prior work [6], we assume that the storage device internally consists of a synthesizer, a PCR thermocycler, and a sequencer as shown in Figure 1. However, in prior work, the storage device is responsible for doing both encoding and synthesis during a put operation, and decoding and resequencing during a get operation.

In OligoArchive, we make a clear division of labor between the DNA storage device and the database engine. We expect the DNA storage system only to provide sequencing and synthesis functionality. The encoding and decoding of data is done by the database engine. Thus, the DNA storage device in OligoArchive takes as an object identifier a DNA sequencing primer that uniquely identifies a group of oligos. The object retrieved is a file containing the textual representation of oligonucleotides that encode the data corresponding to that primer. Internally, the storage system uses the synthesizer to handle the put operation by using the primer and payload oligos to generate synthetic DNA. A get operation is handled by the sequencer that uses the primer to filter and extract all DNA strands that correspond to that primer.

4. DNA DATA STORAGE

In this section, we will focus on the database extensions required for implementing the OligoArchive architecture. We use PostgreSQL as the target database. Thus, we will describe the internals of pg_oligo_dump and pg_oligo_restore, the utility programs we developed to archive data from a PostgreSQL database.

4.1 Pg_oligo_dump: Encoding data

pg_oligo_dump uses a pipeline of several stages to transform relation data stored in a database into synthesizable oligonucleotide sequences as shown in Figure 3.

Extraction and preprocessing. The first stage in the pipeline uses existing support functions in PostgreSQL to extract the data out of the database. Once extracted, dictionary encoding is applied to convert variable length string fields into fixed length integers. In addition to compressing data, this process reduces the record length, making it possible to pack multiple records in a single oligo.

DNA data representation. In the second stage, we use the approach proposed by Goldman et al. [12] to convert the data into oligonucleotide sequences. In this approach, relational data is first converted using a Huffman code into a sequence of ternary digits in base3 format. Each ternary digit is then mapped to a nucleotide using a rotating code. This use of such data representation has been shown to avoid several structural complexity issues like homopolymer repeats and high GC content mentioned in Section 3.

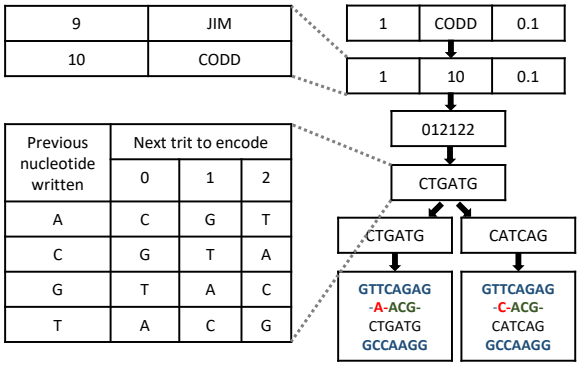

Figure 2: Stages in the data encoding pipeline: Figure shows how a relational tuple is encoded into oligonucleotide sequences. First, dictionary compression is applied to string fields. Second, Huffman coding is used to convert data into trits. Third, a rotating code is used to convert trits into nucleotides. Finally, identification (tableID in green), error detection (parity in red) and correction (reverse-completed mirror) metadata is added together with the primers (blue).

Schema-aware encoding. While the data representation technique is similar to Goldman et al., we differ in the way we drive the encoding process. All current approaches to DNA storage, including the work from Goldman et al., are application agnostic. Thus, they work on binary blobs with no inherent structure. As we mentioned earlier, each oligo is limited to a length of few hundred nucleotides due to synthesis limitations. Thus, each blob is broken into several chunks, and each chunk is encoded in a single oligo. As oligos cannot be explicitly addressed, current approaches also embed addressing information in each oligo in order to identify the index of the chunk in the original blob. Assuming 2 bits per nucleotide, if each oligo has L nucleotides, N nucleotides are reserved for storing addressing information, where N is $log_4 K$ given K, the number of segments. Thus, as K increases, the number of nucleotides available for storing data decreases. For instance, assuming an oligo length of 150 nucleotides, storing 1TB of data requires using 17 nucleotides per oligo for storing addressing information.

We avoid addressing by exploiting database knowledge to perform schema-aware encoding. Each table in a relational database has clearly defined primary keys that can uniquely identify records. Thus, if we encode data such that each oligo maps to a record, no additional addressing information is needed as the logical information already provides structure. However, tables also vary with respect the number of columns and record sizes. Thus, for some tables, it might be possible to encode more than one record in an oligo, while for others, a record might be too large to fit in a single oligo. Thus, our encoding scheme treats these cases separately.

The encoder first performs a preanalysis to find the maximum record size for any given table. Each table is assigned a unique ID which is encoded in all oligos belonging to that table using three nucleotides. We use three nucleotides as the TPC-H database we encode has nine files, namely, eight tables and one dictionary. For tables with records that fit in a single oligo, the encoder iterates over the records and greedily groups records such that all grouped records fit in a single oligo. Thus, a record never crosses an oligo boundary and these tables do not need any further addressing information. For tables with records that are too large to fit in a single oligo, the encoder performs vertical fragmentation by decomposing the table into column sets, and stores one column set per oligo. In order to match a column set with its peers during data restoration, the encoder also adds the primary key column to each column set.

This encoding reduces the addressing overhead from $log_4 K$, where K is the number of segments, to $log_4 C$, where C is the table cardinality. Finally, for the dictionary, the encoder uses the traditional approach by breaking it into chunks that are explicitly indexed.

Error correction metadata. After data has been encoded, the encoder also adds additional metadata for error correction. The encoder adds a parity nucleotide to each oligo to detect errors that might have been introduced during the synthesis or sequencing process. To recover from such errors, the encoder also mirrors each oligo by duplicating and reverse complementing it (reversing the order and switching Gs/As with Ts/Cs and vice versa). As we show later, this coverage was sufficient to guarantee full recovery in our experiments. Finally, sequencing primers are added to either end of each oligo and the oligo string is written out to a file.

4.2 Pg_oligo_restore: Decoding data

Pg_oligo_restore starts with two files containing all the paired-end reads produced by the sequencer of the DNA storage system. First, it uses well-known algorithms for merging the forward and reverse reads into a single consensus read-assembly file [30]. For each oligo in the read assembly file, it performs a validity check based on the expected oligo length and parity, and removes invalid oligos. As pg_oligo_encode stores each oligo twice both in forward and reverse complemented form, pg_oligo_restore attempts to pair each oligo with its reverse complement, discarding oligos that do not have a corresponding pair. It groups oligos into bins based on their table ID and processes each table separately depending on the type of encoding used.

For each oligo, the rotating code is used to convert nucleotides into ternary trits in base3 format, and the Huffman code is used to restore back data. For tables with records spanning multiple rows, the primary key field in each column set is used to combine them back into a single record. In doing so, pg_oligo_restore uses schema information to identify the correct ordering of column sets. The dictionary is recovered by assembling the chunks in the correct order based on the indexing information. Finally, pg_oligo_restore uses the dictionary to restore back the originally record with variable length string fields, and uses existing data importing facilities available in PostgreSQL to import data back into the database.

5. IN-VITRO QUERY PROCESSING

So far, we investigated extensions to the database engine for enabling encoding and decoding of data on DNA. However, DNA, and the mechanisms used to manipulate it, have also been used for computation in a handful of case studies, e.g., for the Hamiltonian path problem [1] or the strategic assignment problem [27]. Some efforts have even tried to replicate logical gates using DNA polymerase [5] to build a transcriptor (biological transistor) to build DNA chips for general purpose computing (within cells). DNA, as a computing medium, does not provide any fundamentally new computational capability, nor is manipulation of one oligionucelotide particularly fast. Its promise, however, lies in the unprecedented parallelism, i.e., in working on countless oligos in parallel.

In this, work we explore processing data directly in the DNA storage by realizing SQL operations using molecular biology techniques. More precisely, with the help of PCR, DNA nucleases [24] and overlap-directed DNA assembly methods, we can execute selection, projection as well as joins on the records encoded as oligos. The sensitivity of these molecular biology techniques is sufficient to perform operations on single oligos in a background of billions of irrelevant oligos. PCR (and similar isothermal amplification methods), for example, can detect single DNA sequences in a background of 100ng of irrelevant DNA sequences [19]. Depending on how the data is encoded as oligos, this means that PCR can find a single record of a few Bytes in a database of 46 TB (assuming two bits encoded with one nucleotide). Thus, in the context of DNA storage and relational query processing, using DNA computing promises to perform operations orders of magnitude faster thanks to the unprecedented parallelism compared to traditional computing.

Figure 3: Encoding of an attribute of a database record as oligo.

Pushing down operations to DNA storage can deplete the oligos. However, using *in vitro* amplification methods (such as PCR amplification) DNA can efficiently and cheaply be replicated on a massive scale.

5.1 Selection & Projection

Selection. Given our encoding of database records as oligos we can use PCR to find a specific record based on its attribute(s), i.e., we can essentially execute a point query. As discussed previously, the primers are used to indicate the exact beginning and end regions of the DNA strand that will be amplified during PCR. If we can encode the attribute we want to use in the query as amplification primers and flank the remaining attributes with them, we can use PCR and subsequently sequencing to execute a point query as only the oligos matching the primers will be read during sequencing.

PCR, however, is not a precise operation and a primer may also amplify DNA sequences that are significantly similar, but not identical, to the target sequence. Key to successfully performing a selection thus is the design of the encoding of attribute values (and the matching primers). Hence we need to use a different method than the one described in Section 4 to encode attribute values that will enable us to selectively retrieve records.

To enable in-vitro operation over data, we encode each attribute of a record in a separate oligo. In each oligo we encode the table and attribute name, the record ID (the primary key), the attribute value, as well as error detection and correction codes (Reed Solomon). Given the restriction on the length of oligos, and the relative length of the first two fields (table and attribute name as well as the record ID), we calculate a fixed length hash code of either field and encode this using a method similar to Church [18] which encodes two bits as one nucleotide.

While encoding the attribute value, we need to make ensure that similar values of the same attribute are encoded to produce substantially different sequences so that the imprecise molecular cloning techniques that we use to perform an in-vitro join (described below) do not end up matching spurious records. To do so, we use the encoding outlined before but we further calculate the checksum of the attribute value using SHA3 and encode it as well. To then make similar attributes encodings substantially different, we split this encoding of an attribute a as well as the checksum c separately into subsets (a_1, a_2, a_3, ... and c_1, c_2, c_3, ...) and interleave one with the other (resulting in a_1, c_1, a_2, c_2, a_3, c_3, ...). Thanks to the avalanche effect of SHA3 (and other cryptographic functions) — small differences in input value lead to considerable differences in the checksum — the resulting sequence will be substantially different, even if the attribute values are similar.

Finally, we calculate the Reed Solomon Code error correction code over the fields as well as the attribute value, encode it and wedge it in between the encoded field and value pairs. Figure 4 illustrates the complete structure of an oligo encoding one attribute

Figure 4: DNA assembly used to splice/join oligos encoding database records.

of a record, with the hash code of the fields on the left, attribute value on the right, and error correction metadata in the middle.

Projection. Given how short the oligos encoding records are, the overhead of sequencing unneeded attributes from oligos does not introduce undue overhead. To implement projection, we thus use PCR to amplify and retrieve specific attributes (as before). These PCR products are sequenced to perform the projection *in silico*.

5.2 Join

Performing a join on records encoded in DNA can be accomplished using molecular cloning techniques as well. At their core, molecular cloning or DNA assembly methods cut and anneal single-stranded DNA sequences where the sequence of their nucleotides is exactly complementary. Cells use these mechanisms to defend against viruses, and to repair broken DNA molecules, while biologists have re-purposed the natural enzymes involved to create synthetic DNA sequences encoding complex biological behavior.

Multiple overlap-directed DNA assembly methods exist [9], but on a conceptual level they work similarly: two single-stranded DNA molecules are spliced together if they match, i.e., if subsequences of nucleotides are complementary. This mechanism allows us to implement an efficient equi-join if the attribute values are the same (and are encoded equally) and their DNA sequences are thus complementary. Overlap-directed DNA assembly processes have a similar specificity as PCR does and can thus join two records/oligos in a background of Terabytes of data.

Figure 4 illustrates the join based on DNA assembly. Two records represented by two oligos (different colors of the different parts of the oligos indicate different values of the fields) are, if the attributes match, spliced together. PCR is then used to amplify the result of the join before it is sequenced. Only annealed/joined oligos are amplified as we use the ends of the single oligos (the pink and purple ends) as primers for PCR.

Similarly to PCR, encoding the attributes must be done such that similar, but non-identical, values result in considerably different DNA sequences. We therefore use the same encoding of join attributes as before to avoid spurious values from being joined. Such approximate joins can be an interesting approach for applications other than equi-joins, but they require further research.

6. EXPERIMENTAL EVALUATION

To demonstrate the feasibility of OligoArchive, we performed an end-to-end experiment where we archived data from a PostgreSQL v10.3 database to synthetic DNA and restore it back again. In this section, we provide details about various experimental stages together with preliminary results that show that DNA can indeed be integrated into the database storage hierarchy.

6.1 Encoding and Synthesis

Due to economic considerations, we limited ourselves to a TPC-H SF 10^{-4} dataset. The input data consists of 44 records stored across eight tables with a total size of 12KB. After loading the data into a PostgreSQL database, we use pg_oligo_dump to generate oligos of 138 nucleotides each. Of the 138 nucleotides, data is encoded in 91 nucleotides and the remainder is used by 5' and 3' primers that use 26 and 21 nucleotides respectively. Table 1 shows

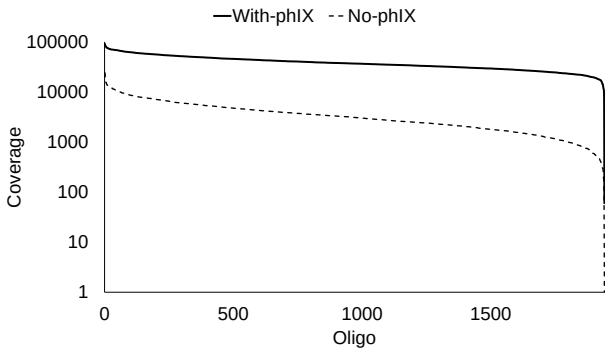

Figure 5: Sequencing coverage: Figure shows the number of times each oligo is read perfectly in our sequencing experiment.

the input cardinalities and the number of oligos generated for each table. The records from the first five tables fit in a single oligo which pg_oligo _dump generates in addition to its reverse complemented copy. Nation table is an example where pg_oligo_dump was able to pack multiple records in a single oligo, and lineitem table represented the case where each record was too large to fit in one oligo and was thus partitioned into two columnsets. In addition, we also encoded the string dictionary which generated 346 oligos, and two images (for other experimental purposes) to generate 1547 oligos. Thus, we synthesized a total of 1941 oligos of 150 nucleotides covering 12KB of TPC-H data and two images. The resulting dried down oligo pool had a weight of 103ng.

Table	Cardinality	#oligos
customer, part, orders, supplier, region	1	2
partsupp	4	8
nation	25	16
lineitem	6	24

Table 1: Input cardinality and oligo counts for TPC-H tables.

6.2 Sequencing and Restoration

The synthesized oligos were prepared for sequencing by amplification via PCR using 10ng of the pool. Illumina NextSeq 500 platform was used for resequencing the library which produced 625 million reads. An analysis of these reads revealed that only 2 million of these reads were error free reads, indicating the existence of large number of errors in the sequencing process. On analyzing the reads, we found the existence of low nucleotide diversity across reads, particularly in the metadata region of oligos, contributed to an increase in sequencing errors. In order to improve diversity, we repeated sequencing with PhiX spike-in. Figure 5 shows the sequencing depth, or the number times each oligo is covered, across the 1941 sequences for both cases, with and without PhiX. Clearly, the PhiX spike-in substantially improves the overall quality of reads as evidenced by 10× to 100× improvement in oligo coverage. Second, as expected, coverage is heavily skewed. While some oligos are nearly hundreds of thousands of times, there is one oligo which was covered only 60 times with PhiX and just once without PhiX.

Table 2 shows additional statistics about each stage of the post-processing phase for the sequencing run with spike-in. Sequencing produced an intial set of 388 million reads. Filtering these reads based on length (91nt) and quality (no spurious base calls) produced 87 million reads. Filtering further based on file id produced 19 million reads corresponding to the Postgres oligos, 65 million for the images, and 3 million reads with invalid file id due to synthe-

sis or substitution errors. Filtering these unique reads based on the pairing requirement as explained in Section 4.2 produced 43,925 unique reads. After performing parity and decode validation, we finally retained 1,281 reads. Although these reads cover the original 404 oligos, we had several erroneous reads that were minor variations of the original oligos with nucleotide substitutions. In spite of such errors, schema information was used to easily identify erroneous data and perform a full recovery and restoration of the original database automatically.

Stage	#Reads
Initial	388M
Length & quality filter	87M
ID filter	19M
Pair filter	43,925
Parity filter	23,409
Decode filter	1,281

Table 2: Post-processing statistics.

Although we were able to fully recover all data using our decoder due to very high sequencing coverage, further analysis is required to understand the scalability of our decoding process for both larger database sizes, and lower degrees of sequencing coverage. For instance, consider the oligo that was covered only once in the sequencing run without PhiX. Had that oligo not been covered during sequencing, our decoder would have failed to automatically recover the data as it eliminates oligos that do not have a matching pair. On analyzing the reads, we found that the reverse complemented pair of this oligo was covered 1700 times. Thus, manual introspection would have still been able to recover the data. However, such human-in-the-loop techniques do not work at scale. Thus, we are investigating the use of clustering algorithms for grouping oligos into similar sets and developing consensus based on approximate string matching. An interesting twist in our case is the fact that we can apply schema-aware clustering algorithms to group records or attributes compared to prior work that uses schema-agnostic ones to cluster raw reads [23].

6.3 In-vitro Query Processing

In the following we describe the experiments for selection and join push down to DNA.

Selection. In the selection experiment, we want to study the sensitivity of the method, i.e., given a query with high selectivity, can we find and retrieve the matching oligo(s)?

We use two different records/oligos from the supplier table, one in low concentration (a few of the same oligo) and one in high as well as increasing concentration (massive number of the same oligo). We add the high concentration to emulate the background, i.e., irrelevant records that do not match the query (ideally these would be different oligos but encoding the whole of the doing so is costly). We encode the two records with the approach discussed in Section 5. The resulting linear oligos are then amplified using PCR. We used Nanopore sequencing [26] (see Appendix A for the protocol) to read out a specific oligo encoding the partkey attribute of the partsupp table in a background of random oligos (same length).

Join. In the join experiment we combine matching oligos, i.e., records with the same attributes, against a background of oligos with randomised sequences. As we are interested in the sensitivity of the method, we use few matching oligos in massive counts as well as increasing number of non matching ones. More specifically, we perform a join between one oligo/record from the parts and partsupp tables using the scheme described in section 5. The

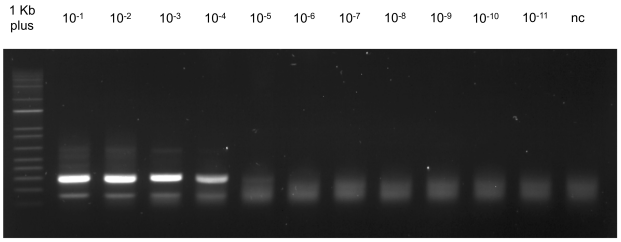

Figure 6: Oligo join sensitivity: specific oligo amplification diluted at different ratios in a background of random oligos. Amplification bands can be seen up to 10^{-5} dilution, meaning that two specific oligos are annealed/joined in a background of 10^5 random oligos.

background oligos had the same length together with the same table and attribute names, however the primary key, error correction and the fixed length hash value were randomised (by ordering these positions as degenerate nucleotides). The initial annealing reaction was conducted using known primers that specifically anneal to each other (namely oligos encoding the partkey attributes of both, the partsup and parts table) combined in a mixture with a pool of primers with randomized bases. The protocol of the experiment is summarised in Appendix A.

The results of the sensitivity analysis are shown in Figure 6 illustrated by gel electrophoresis (after the amplification step). As the figure shows, the join result can be seen up to a dilution of 10^{-5} meaning that one pair of matching oligos joins in a background of 10^5 irrelevant or non-matching oligos. Using Nanopore sequencing (the detailed protocol is in Appendix A), we have also been able to read out the correct join results by sequencing from the join reaction.

7. CONCLUSION

Driven by recent advances in the biotechnology industry, computer architects have recently argued that time is ripe for integrating biomolecules as an integral part of system design [6]. In this paper, we take the first step from the perspective of data management systems towards this goal by presenting OligoArchive–an architecture for using DNA as the archival tier of a relational DBMS. Our experiments show that it is not only feasible to archive and restore data using synthetic DNA, but also exploit database knowledge for optimizing the encoding and decoding process, and even execute SQL operations directly over DNA. In doing so, we set the stage for a new breed of data management systems that are not just biological inspired, but biologically powered.

ACKNOWLEDGMENTS

We thank France Génomique (ANR-10-INBS-09-03, ANR-10-INBS-09-02) for the sequencing support.

8. REFERENCES

[1] L. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 266(5187), 1994.

[2] R. Appuswamy, R. Borovica-Gajić, G. Graefe, and A. Ailamaki. The Five-minute Rule Thirty Years Later and its Impact on the Storage Hierarchy. In *ADMS*, 2017.

[3] S. Balakrishnan, R. Black, A. Donnelly, P. England, A. Glass, D. Harper, S. Legtchenko, A. Ogus, E. Peterson, and A. Rowstron. Pelican: A Building Block for Exascale Cold Data Storage. In *OSDI*, 2014.

[4] J. M. S. Bartlett and D. Stirling. A History of the Polymerase Chain Reaction. *PCR Protocols*, 2003.

[5] J. Bonnet, P. Yin, M. E. Ortiz, P. Subsoontorn, and D. Endy. Amplifying Genetic Logic Gates. *Science*, 340(6132), 2013.

[6] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss. A DNA-Based Archival Storage System. In *ASPLOS*, 2016.

[7] R. Borovica-Gajić, R. Appuswamy, and A. Ailamaki. Cheap Data Analytics using Cold Storage Devices. In *VLDB*, 2016.

[8] R. Carlson. Time for new dna synthesis and sequencing cost curves. http://www.synthesis.cc/2014/02/time-for-new-cost-curves-2014.html, 2014.

[9] A. Casini, M. Storch, G. S. Baldwin, and T. Ellis. Bricks and Blueprints: Methods and Standards for DNA Assembly. *Nature Reviews Molecular Cell Biology*, 16(9), 2015.

[10] Y. Erlich and D. Zielinski. DNA Fountain enables a robust and efficient storage architecture. *Science*, 355(6328), 2017.

[11] R. Fontana. A Ten Year Storage Landscape LTO Tape Media, HDD, NAND. http://storageconference.us/2018/Presentations/Fontana.pdf, 2018.

[12] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney. Toward Practical High-capacity Low-maintenance Storage of Digital Information in Synthesised DNA. *Nature*, 494, 2013.

[13] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angew. Chem. Int. Ed.*, 54, 2015.

[14] A. Hadian and T. Heinis. Towards Batch-Processing on Cold Storage Devices. In *ICDE*, 2018.

[15] Imagene. DNA Shell and RNA Shell. http://www.imagene.eu/.

[16] Intel. Cold Storage in the Cloud: Trends, Challenges, and Solutions. White Paper.

[17] S. Kosuri and G. M. Church. Large-scale de novo DNA synthesis: technologies and applications. *Nature Methods*, 11(5), 2014.

[18] H. H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. M. Church. Enzymatic dna synthesis for digital information storage. *bioRxiv*, 2018.

[19] T. Notomi, H. Okayama, H. Masubuchi, T. Yonekawa, K. Watanabe, N. Amino, and T. Hase. Loop-mediated isothermal amplification of dna. *Nucleic Acids Research*, 28(12):e63, 2000.

[20] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss. Random access in large-scale DNA data storage. *Nature Methods*, 11(5), 2014.

[21] PCWorld. Facebook uses 10,000 blu-ray discs to store 'cold' data. http://www.pcworld.com/article/2092420/facebook-puts-10000-bluray-discs-in-lowpower-storage-system.html.

[22] M. Perlmutter. The Lost Picture Show: Hollywood Archivists Cannot Outpace Obsolescence. https://spectrum.ieee.org/computing/it/the-lost-picture-show-hollywood-archivists-cant-outpace-obsolescence, 2017.

[23] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss. Clustering billions of reads for dna data storage. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NIPS*. 2017.

[24] R. J. Roberts and K. Murray. Restriction Endonuclease. *CRC Critical Reviews in Biochemistry*, 4(2):123–164, 1976.

[25] D. Rosenthal. The Medium Term Prospects for Long Term Storage. https://blog.dshr.org/2016/12/the-medium-term-prospects-for-long-term.html, 2016.

[26] G. F. Schneider and C. Dekker. DNA Sequencing with Nanopores. *Nature Biotechnology*, 30:326, apr 2012.

[27] J.-J. Shu, Q.-W. Wang, and K.-Y. Yong. DNA-Based Computing of Strategic Assignment Problems. *Phys. Rev. Lett.*, 106, May 2011.

[28] W. Yan, J. Yao, Q. Cao, C. Xie, and H. Jiang. ROS: A Rack-based Optical Storage System with Inline Accessibility for Long-Term Data Preservation. In *EUROSYS*, 2017.

[29] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic. A Rewritable, Random-Access DNA-Based Storage System. *Nature Scientific Reports*, 5(14318), 2015.

[30] J. Zhang, K. Kobert, T. Flouri, and A. Stamatakis. PEAR: a fast and accurate Illumina Paired-End reAd mergeR . *Bioinformatics*, 30(5), 2013.

APPENDIX

A. EXPERIMENTAL PROTOCOL JOIN

Primers were annealed to one another and extended in a single 20 µL reaction. This mixture consisted of 13 µL of Milli-Q water, 4 µL HF Buffer, 0.0 - 0.9 µL of each specific primer pair (parts-partkey, parts-partkey-RC, partsup-partkey and partsup-partkey-RC), 0.0 - 0.9 µL of each random primer pair (parts-random and partsup-random) and 0.2 µL of Phusion polymerase. The reaction was performed in an Applied Biosystems Veriti 96 well thermal cycler, programmed for denaturation at 98 °C for 30 seconds, cooled down to 63 °C over 1 hour, and annealed at 63 °C for 10 mins, followed by extension at 72 °C for 15 mins. We added random primers in varying proportions (summing to 0.9 µL) to test sensitivity. All primers and dNTPs are from 10 µM stock.

The product of the annealing reaction was then used in the subsequent amplification step, using the primers partsup-amp and parts-amp. The 20 µL mixture consisted of 8.4 µL Milli-Q water, 4 µL HF buffer, 0.4 µL dNTPs, 5 µL PCR mixture from the annealing step, 1 µL of partsup-amp, 1 µL of parts-amp, and 0.2 µL of Phusion polymerase. The thermal cycler was programmed at 98 °C for initial denaturation, followed by 30 cycles of the following: denaturation at 98 °C for 10s, annealing at 61.5 °C for 30s and elongation at 72 °C for 15s. This was then followed by 10 minutes of a final extension at 72 °C.

Oxford Nanopore DNA sequencing was carried out according to the manufacturer's instructions, following the 1D amplicon/cDNA by ligation protocol using the SQK-LSK109 sequencing kit. The only change made was the omission of the NEBNext FFPE DNA Repair Buffer, which was replaced by Ultra II End-prep reaction buffer, and the NEBNext FFPE DNA Repair Mix, which was replaced by nuclease free water.