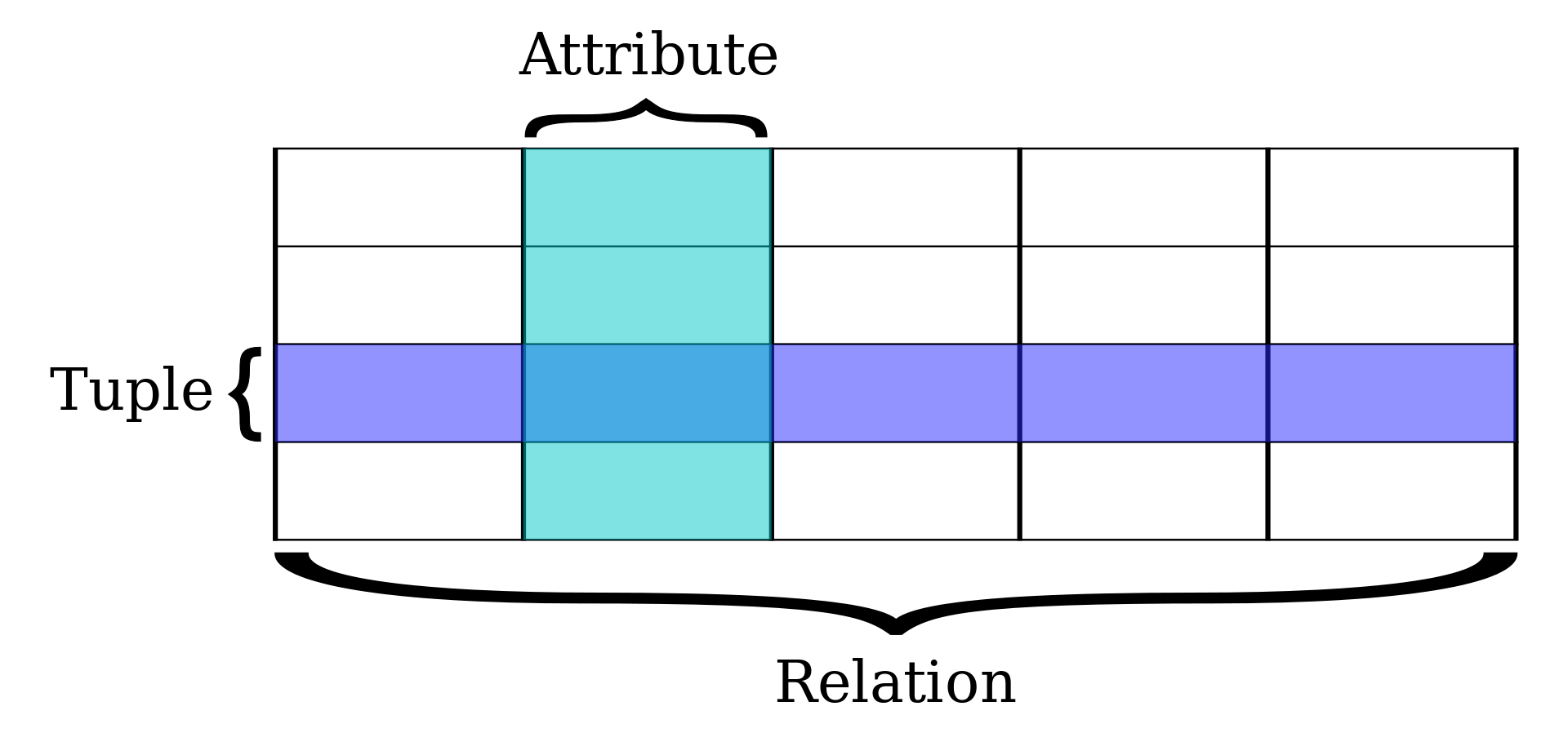
When considering the correct database for our project, we had the option of choosing between four options: the standard relational databases, non-relational databases, or go with a left field choices such as data warehouses or a solution which is radically gaining popularity, blockchain. To come to a final decision, we have decided to list the pros and cons of each solution to make an informed final decision to see which approach fits our needs the most.

Relational databases are great at organizing and retrieving structured data. All data is stored and accessed with the help of relations (usually described as tables). This is the most widely used solution as its very robust and great for fast transactions and is our choice for the project.



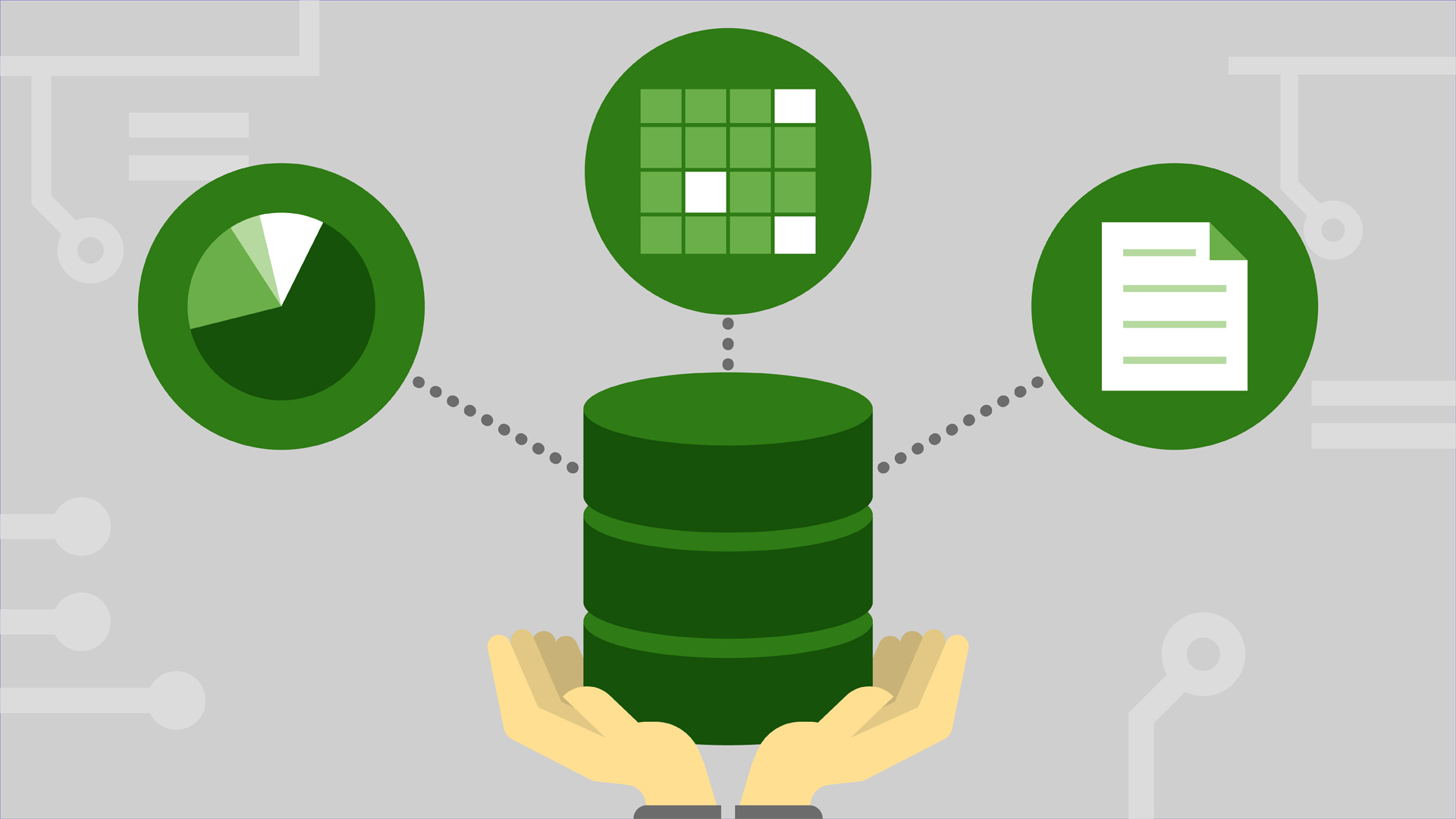
**Pros**

* Relational databases work with structured data.
* They support ACID transactional consistency and support “joins.”
* They come with built-in data integrity and a large eco-system.
* Relationships in this system have constraints.
* There is limitless indexing. Strong SQL.

**Cons**

* Relational Databases do not scale out horizontally very well, only vertically.
* Data is normalized, meaning lots of joins, which affects speed.
* They have problems working with semi-structured data.

Non-relational databases or NoSQL are best applicable when data is inconsistent, incomplete or its amount is massive, this approach is very popular in the bigdata field. The top NoSQL database engine, which we are familiar with, and is a document store database, would be MongoDB. NoSQL databases are also a great option for the project as they scale better (better results without the need of upgrading existing hardware) and are more flexible (can accommodate different types of documents as there is no set schema) compared to traditional SQL databases.



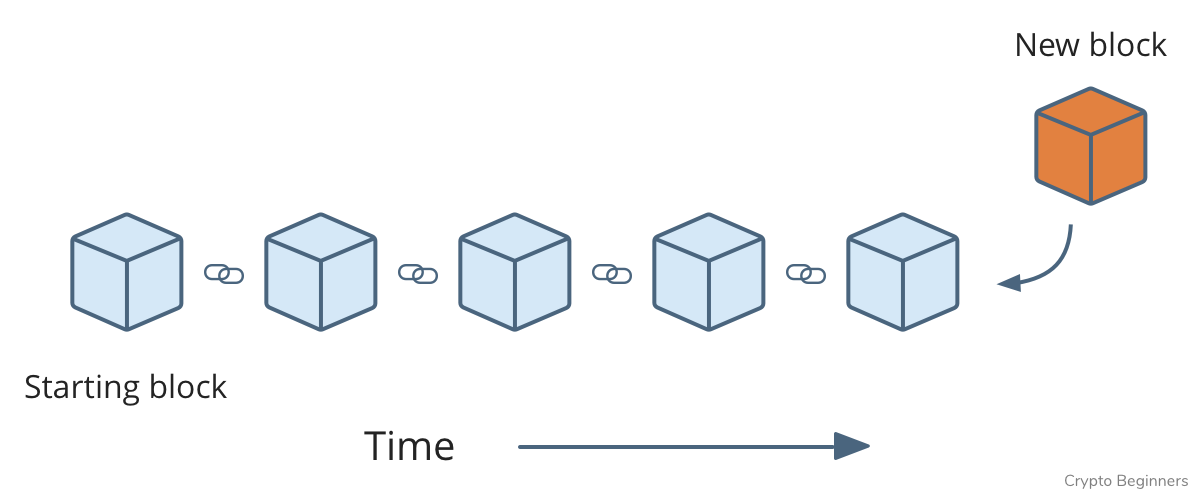
**Pros**

* They scale out horizontally and work with unstructured and semi-structured data.
* Schema-free or Schema-on-read options.
* High availability.
* Many NoSQL databases are open source and so “free”.

**Cons**

* Weaker or eventual consistency (BASE) instead of ACID.
* Limited support for joins.
* Data is denormalized, requiring mass updates.
* Does not have built-in data integrity.
* Limited indexing.
* Requires considerable training, to be used properly, although quite easy for simple use cases.

Blockchain, while a radically different approach to the traditional SQL/NoSQL solution it would be interesting to see how this innovative way of storing data could bypass the need for a future system administrator and open up possibilities to create decentralized applications or Dapps for short, which members of GTL could use to track the status of books or even be incentivized to borrow books in the form of rewards.



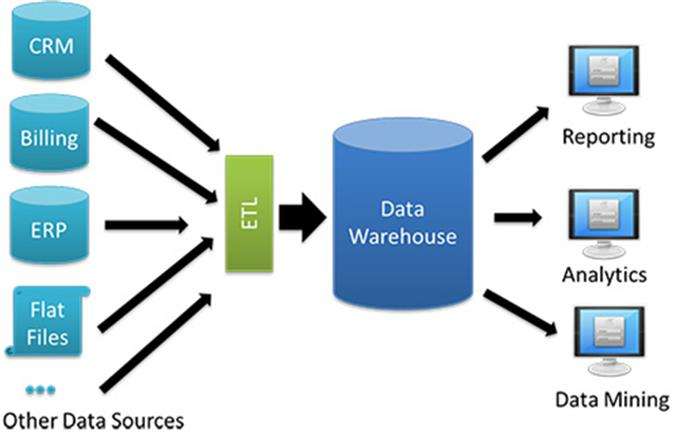
**Pros**

* Transactions are immutable
* All transactions are linked to one and other and are stored forever, great for the statistics/analysis requirement given by GTL
* High availability
* Increased transparency over traditional databases

**Cons**

* Can be costly depending on the type of implementation
* Low interoperability with traditional databases
* No experience with the technology

Data warehouses compared to regular databases use a different design, the latter focuses on being optimized for strict accuracy by being able to rapidly update real-time data. Compared to the fast response nature, of operational databases, data warehouses are designed to be a non-volatile alternative, trading transaction volume to data aggregation and are designed to scale and are great for analytics. We decided to not choose Data warehouses as we need to focus on handling transactions when monitoring books and handling users (although a great solution for the analytic requirements of the project)



**Pros**

* Great for analytics on transactional data for businesses with high IT resources
* Error identification and correction, as data warehouses show inconsistencies and correct them before loading data
* Increased query performance (when talking about large volumes of data) and is easier to extract information

**Cons**

* Preparation is often very time consuming as considerable efforts are needed to create a quality storage solution with properly compatible storage and retrieval
* Security flaws, as
* Needs regular maintenance regularly which can provide a costly downtime