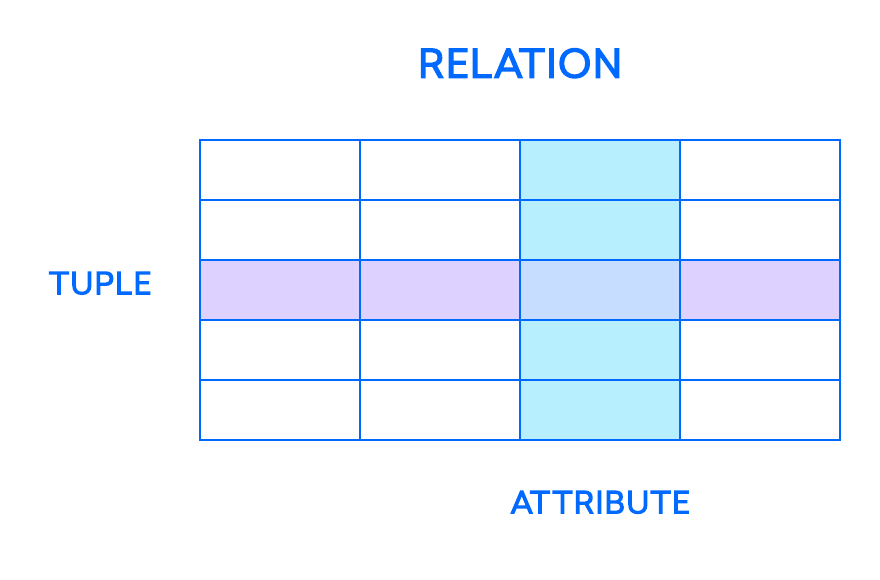
**SQL: how to use it in relation to Dune**

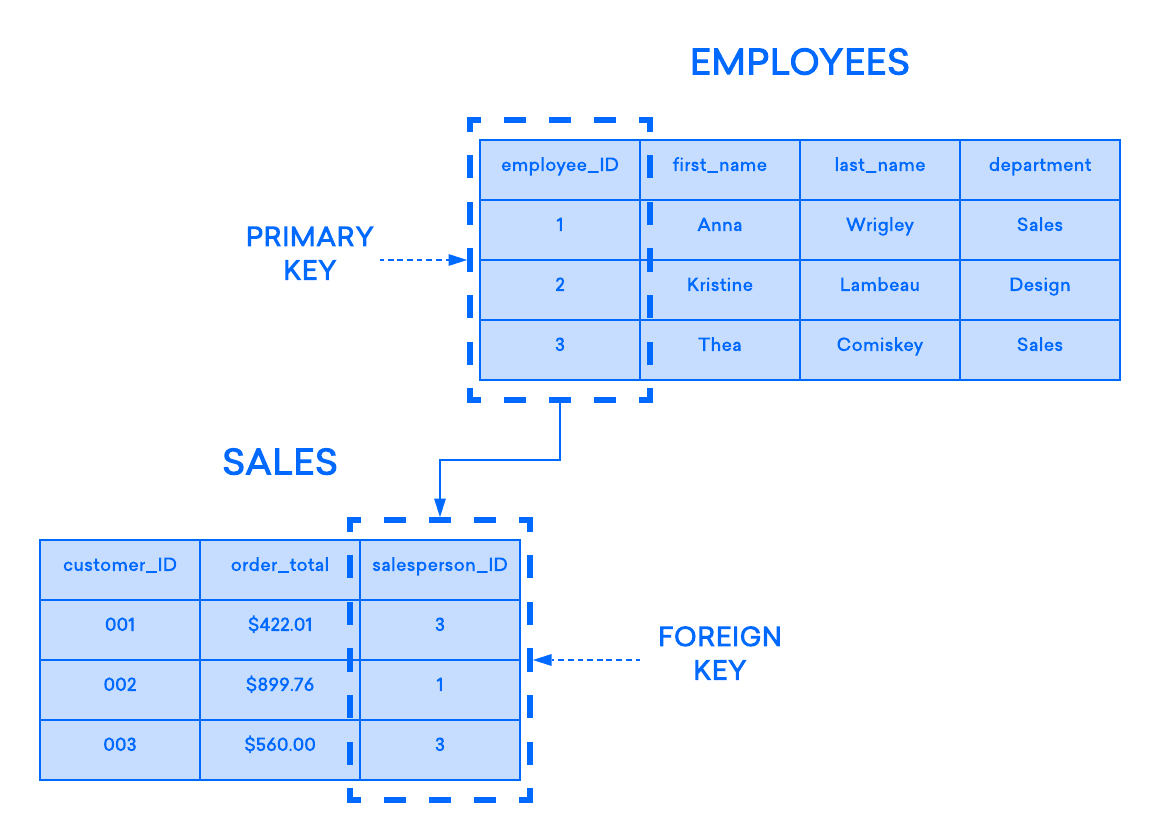
<https://www.digitalocean.com/community/tutorials/understanding-relational-databases>

relational databases:



The most fundamental elements in the relational model are *relations*, which users and modern RDBMSs recognize as *tables*. A relation is a set of *tuples*, or rows in a table, with each tuple sharing a set of *attributes*, or columns:

In the relational model, each table contains at least one column that can be used to uniquely identify each row, called **a primary key**. This is important, because it means that users don’t need to know where the data is physically stored on a machine. Instead, their DMBS can keep track of each record and return them on an ad hoc basis.

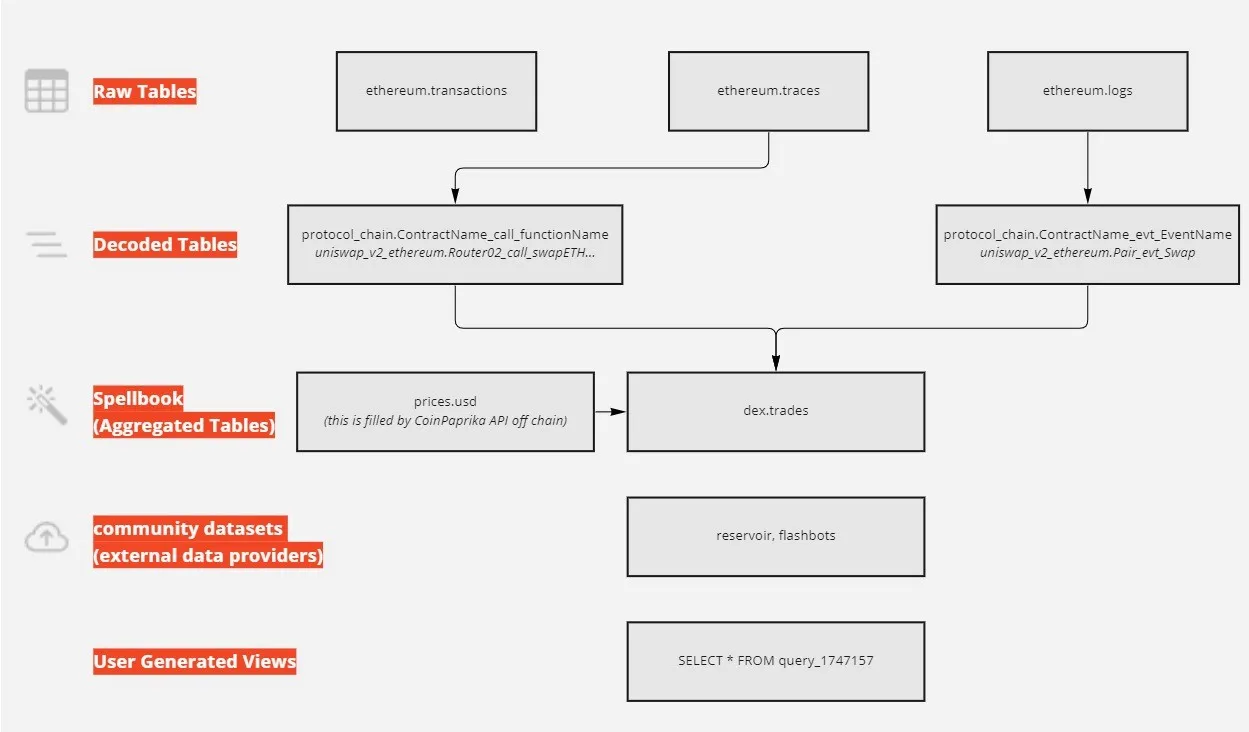
If you have two tables that you’d like to associate with one another, one way you can do so is with a ***foreign key***. A foreign key is essentially a copy of one table’s (the “parent” table) primary key inserted into a column in another table (the “child”). The following example highlights the relationship between two tables, one used to record information about employees at a company and another used to track the company’s sales. In this example, the primary key of the **EMPLOYEES** table is used as the foreign key of the **SALES** table:

**A Basic Wizard Guide to Dune SQL and Ethereum Data Analytics**

<https://web3datadegens.substack.com/p/a-basic-wizard-guide-to-dune-sql>

How the EVM translates to data tables in Dune

* Every transaction is like a row on an excel sheet.
* Tables in dune all have shared/related columns, most commonly being a contract/wallet address or transaction hash.

Hierarchy of tables:

Transaction table:

* From: wallet signing the trasaction
* To: address interacted with
* Input: data passed to call a function

1. Raw tables: transactions, traces, logs and blocks tables representing blockchain data in its rawest form: mostly byte arrays.

* Transaction – a contract (e.g. sending someone some ETH).
  + Each transaction occurs in specific blocks.
  + Transactions have an index denoting the order in which a transaction is executed in the block: **tx\_index**
* A transaction will set off **traces** (contracts calling other contracts, deploying a contract, sending ETH somewhere else).
  + Trace addresses: **trace\_adress**
* Those functions calls can emit events as they execute (called **logs**). Logs are ordered across the block based on index
  + Column is called: **evt\_index** outside of the logs table.

1. Decoded tables:

* Based on contract ABI’s (Application Binary interfaces) submitted to a contracts table (i.e. Ethereum.contracts), functions and events are converted to bytes signatures (Ethereum.signatures) which are then matches against traces and logs to give you decoded tables such as **uniswap\_v2\_ethereum.Pair\_evt\_swao** which stores all swaps for all pair contracts created by the pair factory (you can filter for a specific one by looking at the contract\_adress table for events.
  + Each function and event gets its own table. Read functions will show up buy can’t be queried (i.e. table will be empty for something like balanceOf).
  + All decoded tables carry down the main transaction metadata columns such as tx\_has, block\_time, and block\_number with a call or evt prefix.

After:

<https://web3datadegens.substack.com/p/how-to-start-analyzing-any-web3-protocol>

Finally: <https://web3datadegens.substack.com/p/how-to-analyze-bitcoin-data-with>

Lastly: <https://substack.com/inbox/post/103381870>