**What is NoSQL data base?**

NoSQL encompasses a wide variety of different database technologies that were developed in response to the demands presented in building modern applications:

* Developers are working with applications that create massive volumes of new, rapidly changing data types — structured, semi-structured, unstructured and polymorphic data.
* Long gone is the twelve-to-eighteen month waterfall development cycle. Now small teams work in agile sprints, iterating quickly and pushing code every week or two, some even multiple times every day.
* Applications that once served a finite audience are now delivered as services that must be always-on, accessible from many different devices and scaled globally to millions of users.
* Organizations are now turning to scale-out architectures using open source software, commodity servers and cloud computing instead of large monolithic servers and storage infrastructure.

Relational databases were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the commodity storage and processing power available today.

**How does data get stored in NoSQl database?**

In addition, it provides an "extended relational data modes" such as inheritance between classes, collection data types (SET, MULTISET, LIST) and composition relation.

Inheritance is a concept that allows reusing in child tables the columns and methods defined in parent tables. CUBRID supports inheritance for reusability. By using inheritance feature provided by CUBRID, you can create a parent table with some common columns and then create child tables inherited from the parent table with some unique columns added. This way, you can model a database minimizing the number of columns needed.

However, CUBRID allows the direct use of the physical address (OID) where the records of the referred table are located, so you can define relations without using join operations. That is, in an object-oriented database like CUBRID, you can create a composition relation where one record has a reference value to another by using the column displayed in the referred table as a domain (type), instead of referring to the primary key column from the referred table.

Generally, in object-oriented programs objects are the actual data stored in the memory where object pointers are used to point to those objects. Conversely, CUBRID directly handles the database objects, so it cannot express objects. Instead, it issues a unique Object Identifier (OID) for each object. OID indicates the physical address of a database object, the absolute location in the database volume file on the disk. Like a memory pointer that stands for the physical address in the memory area, an OID is the physical address in the database area.

The OID, the physical address of a database object, consists of a volume number (volid), a page number in the volume (pageid), and a slot number in the page (slotid).

**What is a column family in HBase?**

In the **HBase** data model **columns** are grouped into **column families**, which must be defined up front during table creation. **Column families** are stored together on disk, which is why **HBase** is referred to as a **column**-oriented data store.

Columns in Apache HBase are grouped into *column families*. All column members of a column family have the same prefix. For example, the columns *courses:history* and *courses:math* are both members of the *courses* column family. The colon character (:) delimits the column family from the . The column family prefix must be composed of *printable* characters. The qualifying tail, the column family *qualifier*, can be made of any arbitrary bytes. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up an running.

Physically, all column family members are stored together on the filesystem. Because tunings and storage specifications are done at the column family level, it is advised that all column family members have the same general access pattern and size characteristics.

A {row, column, version} tuple exactly specifies a cell in HBase.

**How many maximum number of columns can be added to HBase table?**

I would say you can have an unlimited number of column families.

**Why columns are not defined at the time of table creation in HBase?**

set of columns that are not dependent on any other columns of other families.

**How does data get managed in HBase?**

NoSQL databases are designed for scalability where unstructured data is spread across multiple nodes. When data volumes increase you just need to add another node to accommodate the growth. The lack of structure in NoSQL databases relaxes stringent requirements of consistency enforced in relational databases to improve speed and agility. Hbase, MongoDB and Cassandra are the three major options that provide NoSQL capabilities.

**What happens internally when new data gets inserted into HBase table?**

To write data to HBase, you use methods of the HTableInterface class. You can use the Java API directly, or use HBase Shell, Thrift API, REST API, or another client which uses the Java API indirectly. When you issue a Put, the coordinates of the data are the row, the column, and the timestamp. The timestamp is unique per version of the cell, and can be generated automatically or specified programmatically by your application, and must be a long integer