Here is an explanation of UUIDs and the best version for database primary keys, based on reliable technical standards (IETF RFCs) and database performance benchmarks.

### What is a UUID?

**UUID** stands for **Universally Unique Identifier**. It is a 128-bit label used to uniquely identify information in computer systems.

* **Format:** It is typically displayed as a 36-character string of hexadecimal digits separated by hyphens (e.g., 550e8400-e29b-41d4-a716-446655440000).
* **Purpose:** The main goal is to allow distributed systems to generate unique IDs without a central authority (like a database counter) coordinating them. You can generate a UUID on a client (like a mobile phone) and trust that it will not collide with a UUID generated on a server halfway across the world.

### Best UUID for Database Primary Key: UUID Version 7

For modern database applications, **UUID v7** is widely considered the best choice for a primary key.

While **UUID v4** (completely random) has been the standard for years, it has major performance drawbacks. **UUID v7** solves these problems by combining a timestamp with randomness.

#### Why UUID v7 is Superior

1. **Database Performance (Index Locality):**
   * **The Problem with v4:** Because UUID v4 is completely random, new records are inserted at random positions in your database index (B-Tree). This causes "fragmentation," forcing the database to constantly reorganize data pages, which destroys write performance as the table grows.
   * **The v7 Solution:** UUID v7 starts with a timestamp (Unix Epoch time). This makes it **sequentially sortable** (monotonically increasing). New records are appended to the *end* of the database index, similar to a traditional auto-increment ID (1, 2, 3...). This drastically reduces database overhead.
2. **Uniqueness:**
   * Like other versions, it still includes enough random bits to guarantee global uniqueness for all practical purposes.
3. **No Privacy Leaks:**
   * Unlike the older **UUID v1**, which embedded your computer's MAC address (physically identifying your machine), v7 uses a timestamp and random numbers, keeping your hardware identity private.

### Comparison of Common Versions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description** | **Use for DB Key?** | **Why?** |
| **v4** | **Random.** The most common version historically. | ⚠️ **Avoid** | It is fully random. It destroys database index performance at scale because it is not sortable. |
| **v7** | **Time-based + Random.** (New Standard RFC 9562). | ✅ **Best** | It is time-ordered (fast for databases) but still distributed and unique. |
| **v1** | **Time + MAC Address.** | ❌ **No** | It relies on your physical MAC address (privacy risk) and the timestamp ordering is often formatted incorrectly for sorting. |

### Summary Recommendation

If your database or programming language supports it, **use UUID v7**.

* **If you are on an older system** that typically generates v4 and you cannot upgrade to v7, you may consider **ULID** (Universally Unique Lexicographically Sortable Identifier), which behaves very similarly to UUID v7 (sortable + random) but uses a different character format.