A set cannot have duplicate elements by its mere definition. The correct structure to allow duplicate elements is [Multiset or Bag](https://en.wikipedia.org/wiki/Multiset):

In mathematics, a multiset (or bag) is a generalization of the concept of a set that, unlike a set, allows multiple instances of the multiset's elements. For example, {a, a, b} and {a, b} are different multisets although they are the same set. However, order does not matter, so {a, a, b} and {a, b, a} are the same multiset.

A very common and useful example of a Multiset in programming is the collection of values of an object:

values({a: 1, b: 1}) //=> Multiset(1,1)

The values here are unordered, yet cannot be reduced to Set(1) that would e.g. break the iteration over the object values.

Further, quoting from the linked Wikipedia article (see there for the references):

Multisets have become an important tool in databases.[18][19][20] For instance, multisets are often used to implement relations in database systems. Multisets also play an important role in computer science.

|  |
| --- |
|  |

Candidate key is a super key from which you cannot remove any fields.

For instance, a software release can be identified either by major/minor version, or by the build date (we assume nightly builds).

Storing date in three fields is not a good idea of course, but let's pretend it is for demonstration purposes:

year month date major minor

2008 01 13 0 1

2008 04 23 0 2

2009 11 05 1 0

2010 04 05 1 1

So (year, major, minor) or (year, month, date, major) are super keys (since they are unique) but not candidate keys, since you can remove year or major and the remaining set of columns will still be a super key.

(year, month, date) and (major, minor) are candidate keys, since you cannot remove any of the fields from them without breaking uniqueness.

**Degree** - number of attributes in a relation (table)

**Cardinality** - number of tuples (rows) present in a table

SQL Injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious *payload*) that control a web application’s database server (also commonly referred to as a *Relational Database Management System – RDBMS*).

By leveraging an SQL Injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application’s authentication and authorization mechanisms and retrieve the contents of an entire database. SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity.

To such an extent, SQL Injection can provide an attacker with unauthorized access to sensitive data including, customer data, personally identifiable information (PII), trade secrets, intellectual property and other sensitive information.

The relational model dictates that a relational database consists of (i) a set of relations and (ii) a set of integrity constraints

– Allconstraintsmet=>databaseinavalidstate • A relation is composed of its schema (name; list of n attributes,

each with its domain) and its state/data (set of n-tuples) • Schema (or explicit) constraints, specified via DDL, include

domain, key, entity integrity, and referential integrity

– Datamanipulationoperations(insert,update,delete;viaDML)canrun awry of these constraints

• A transaction is a sequence of operations and ACID-compliant RDBMSs implement ”proper” transaction processing

– Atomicity,Consistency,Isolation,Durability