Surrogate key

A surrogate key is artificially created in a table when there is no unique identifier and because of that no primary key could have made so in that case this key is generated from database as a unique identifier in the table

Surrogate key vs primary key vs composite key

A primary key is a column or set of columns that uniquely identifies each row in a table. It ensures that each row has a unique identifier and provides a way to uniquely identify and retrieve specific records from the table. The primary key is typically chosen from existing columns in the table that have unique values, such as an ID column.

On the other hand, a surrogate key is an artificially generated identifier assigned to each row in a table. It is not derived from any meaningful or inherent data in the table. Surrogate keys are often implemented as auto-incrementing integers or unique identifiers generated by the database system. The purpose of a surrogate key is to provide a stable and unique identifier for each row, regardless of the actual data values in the table. It is commonly used in situations where there is no natural or suitable primary key available.

In summary, a primary key is chosen from existing data in the table and serves as a unique identifier for each row, while a surrogate key is a generated identifier used when there is no appropriate primary key available.

In the context of database design, a natural key is a column or set of columns that already exist in the data and uniquely identify a record. A composite key is a combination of multiple columns that together form a unique identifier for a record.

A surrogate key, on the other hand, is an artificial or system-generated key that is used as the primary key of a table instead of relying on natural or composite keys. It has no inherent meaning or relationship to the data it represents.

By using a surrogate key, you can eliminate the reliance on existing columns as keys and introduce a simpler and more stable identifier for each record.

How surrogate keys are created[​](https://docs.getdbt.com/terms/surrogate-key#how-surrogate-keys-are-created)

In analytics engineering, you can generate surrogate keys using a hashing method of your choice. Remember, in order to truly create a uniqueness constraint on a database object, you’ll need to hash the fields together that *make each row unique*; when you generate a correct surrogate key for a dataset, you’re really establishing the true [grain](https://docs.getdbt.com/terms/grain) of that dataset.

Let’s take this to an example. Below, there is a table you pull from an ad platform that collects calendar\_date, ad\_id, and some performance columns.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **calendar\_date** | **ad\_id** | **impressions** | **spend** | **clicks** | **conversions** |
| 2022-05-16 | 212 | 88744 | 4523.00 | 9432 | 166 |
| 2022-05-16 | 214 | 323 | 6.49 | 4 | 0 |
| 2022-05-05 | 212 | 125600 | 117244.56 | 17318 | 56 |

In this state, this table has no natural key that can act as a primary key. You know the grain of this table: this is showing performance for each ad\_id per calendar\_date. Therefore, hashing those two fields will create a uniqueness constraint on this table.

To create a surrogate key for this table using the MD5 function, run the following:

select   
 md5(calendar\_date || ad\_id) as unique\_id,  
 \*  
from {{ source('ad\_platform', 'custom\_daily\_report')}}

After executing this, the table would now have the unique\_id field now uniquely identifying each row.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **unique\_id** | **calendar\_date** | **ad\_id** | **impressions** | **spend** | **clicks** | **conversions** |
| 62aef884fbe3470ce7d9a92140b09b17 | 2022-05-16 | 212 | 88744 | 4523.00 | 9432 | 166 |
| ea385f7a5e560ef4d8a78f7d913927e4 | 2022-05-16 | 214 | 323 | 6.49 | 4 | 0 |
| 53a33f257d1d4f2446469ac5adad1c0c | 2022-05-05 | 212 | 125600 | 117244.56 | 17318 | 56 |

Practical implementation

***select \* from DEMO.PRACTICE.SNAPSHOT***

ID NAME SALARY UPDATED\_AT

1 Abdullah Najeeb 135,000 2023-06-05 21:58:53.859

2 Ahmed Rustam 90,000 2023-06-05 21:40:01.610

3 Usama Sajid 90,000 2023-06-05 21:40:03.053

***select md5(id) as unique\_id,\* from DEMO.PRACTICE.SNAPSHOT***

**UNIQUE\_ID ID NAME SALARY UPDATED\_AT**

c4ca4238a0b923820dcc509a6f75849b 1 Abdullah Najeeb 135,000 2023-06-05 21:58:53.859

c81e728d9d4c2f636f067f89cc14862c 2 Ahmed Rustam 90,000 2023-06-05 21:40:01.610

eccbc87e4b5ce2fe28308fd9f2a7baf3 3 Usama Sajid 90,000 2023-06-05 21:40:03.053

**Every model should have some “grain” to it, i.e. what one record represents. For example:**

* customers has one record per customer
* subscription\_periods has one record per subscription, per valid period
* ad\_spend\_by\_campaign\_by\_day has one record per campaign, per day

We use the surrogate\_key macro from [dbt-utils 426](https://github.com/fishtown-analytics/dbt-utils#surrogate_key-source) to generate a primary key based on the grain of a model. For our ad\_spend\_by\_campaign\_by\_day example:

**select**

{{ dbt\_utils.surrogate\_key(

'campaign\_name',

'date\_day'

) }} **as** ad\_spend\_id,

campaign\_name,

platform,

date\_day,

spend,

...

Behind the scenes, this macro hashes the combination of columns, which means that your resulting id will always be the same for a particular record. Or, put another way, the generated ad\_spend\_id is now idempotent.

Conclusion[​](https://docs.getdbt.com/terms/surrogate-key#conclusion)

***Surrogate keys are unique row identifiers that are created by using columns in a database object to create a uniqueness constraint on the data. To create a surrogate key, you will use a cryptographic algorithm usually in the form of the MD5 function to hash together fields that create a uniqueness constraint on the dataset. Ultimately, surrogate keys are a great way to create unique row identifiers for database objects that lack them naturally and allow folks to easily identify the grain of the data.***