DATABASE SECURITY AND DATA MASKING WITH GREENMASK

Vadim Voitenko
Database Engineer | Exness
Co-Founder | Greenmask.IO

Limassol, Cyprus 2024
PERCONA UNIVERSITY

AGENDA

*

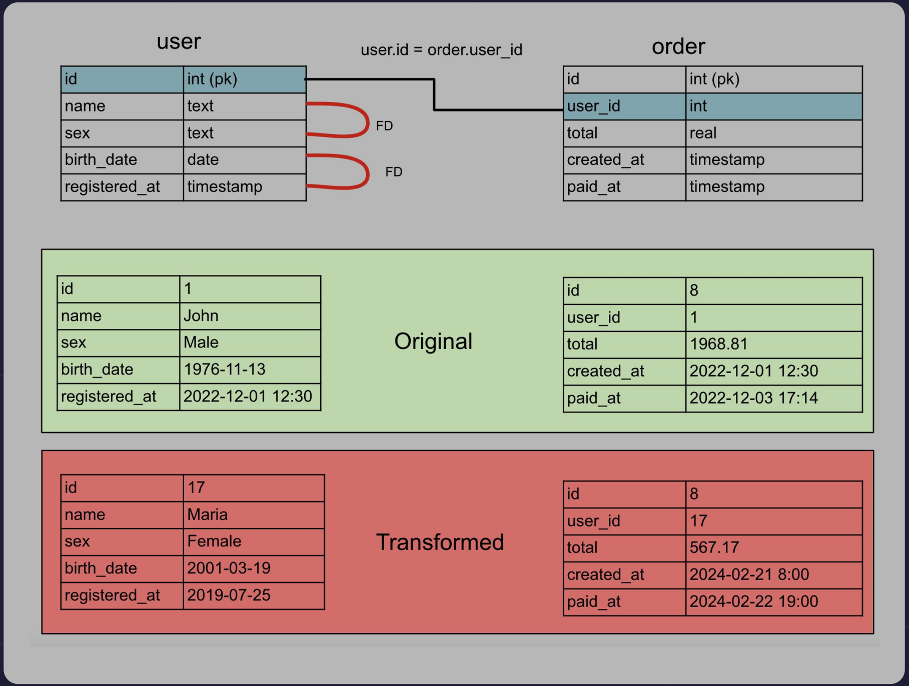
01

DATABASE OBFUSCATION PROCEDURE

WHAT IS DATABASE OBFUSCATION?

REQUIREMENTS:

- Replace original values with fake data
- Keep database consistent
- Subset date (scale down database size)



user

id	int (pk)
name	text
sex	text
birth_date	date
registered_at	timestamp

order

id	int (pk)
user_id	int
total	real
created_at	timestamp
paid_at	timestamp

Original

id	1
name	John
sex	Male
birth_date	1976-11-13
registered_at	2022-12-01 12:30

id	8
user_id	1
total	1968.81
created_at	2022-12-01 12:30
paid_at	2022-12-03 17:14

Transformed

id	17
name	Maria
sex	Female
birth_date	2001-03-19
registered_at	2019-07-25

id	8
user_id	17
total	567.17
created_at	2024-02-21 8:00
paid_at	2024-02-22 19:00

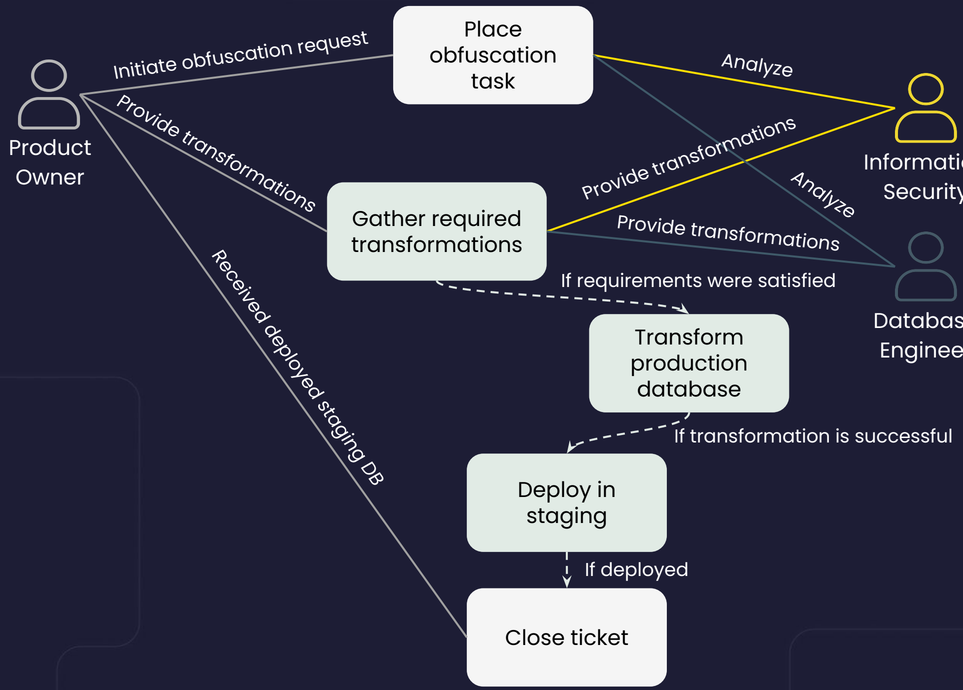
* FD - functional dependency
* PK - primary key

WHO NEED THIS?

- Small to large companies are required in high quality test data
- Outsourcing companies
- Companies uses outsourcing service
- Who deal with machine learning
- Information security and legal entity tribes
- Data engineers and infrastructure teams

DATABASE OBFUSCATION PROCESS

- Involves DBA, IS and Product
- List of sensitive tables and columns
- Transform
- Restore periodically
- Check schema changes

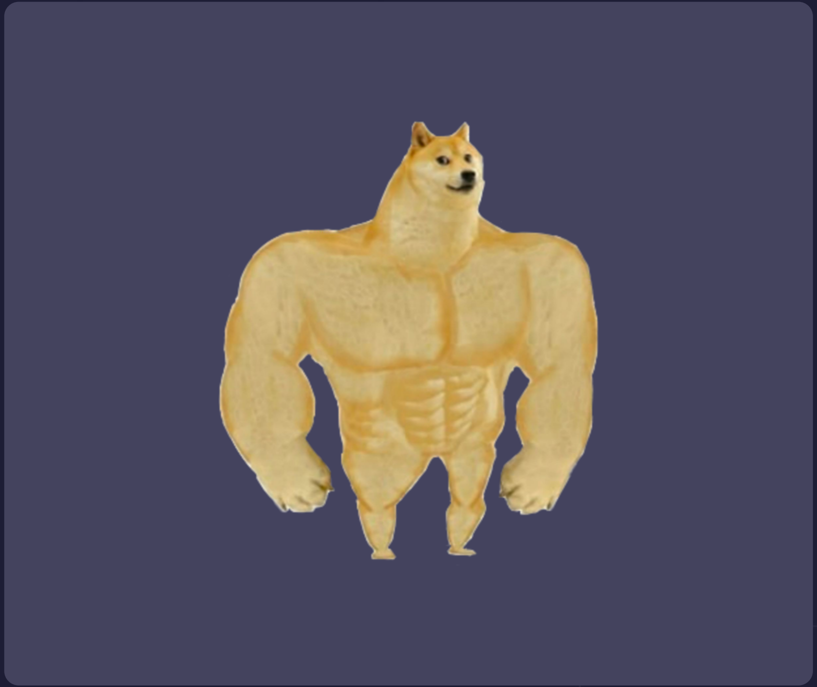


THESES TO REFLECT

- Dump and restore must be reliable
- DB Obfuscation ~ reverse development
- Control schema changes
- Easy extend transformation logic
- Declarative rather than imperative

DESIRED SOLUTION

- Stateless
- Type safe
- Reliable as vendor utilities
- Streaming inline replacement
- Variety of storages
- Customizable
- Interactive (validation and debugging)
- Declarative and integratable



*

GREENMASK DESIGN

GREENMASK. WHAT IS IT?

- Database dumping and anonymization tool
- Written in Go
- Open source (Apache 2.0 License)
- Published 5 months ago
- Backward compatible with vendor utilities
- Easy up and running
- Has playground sandbox

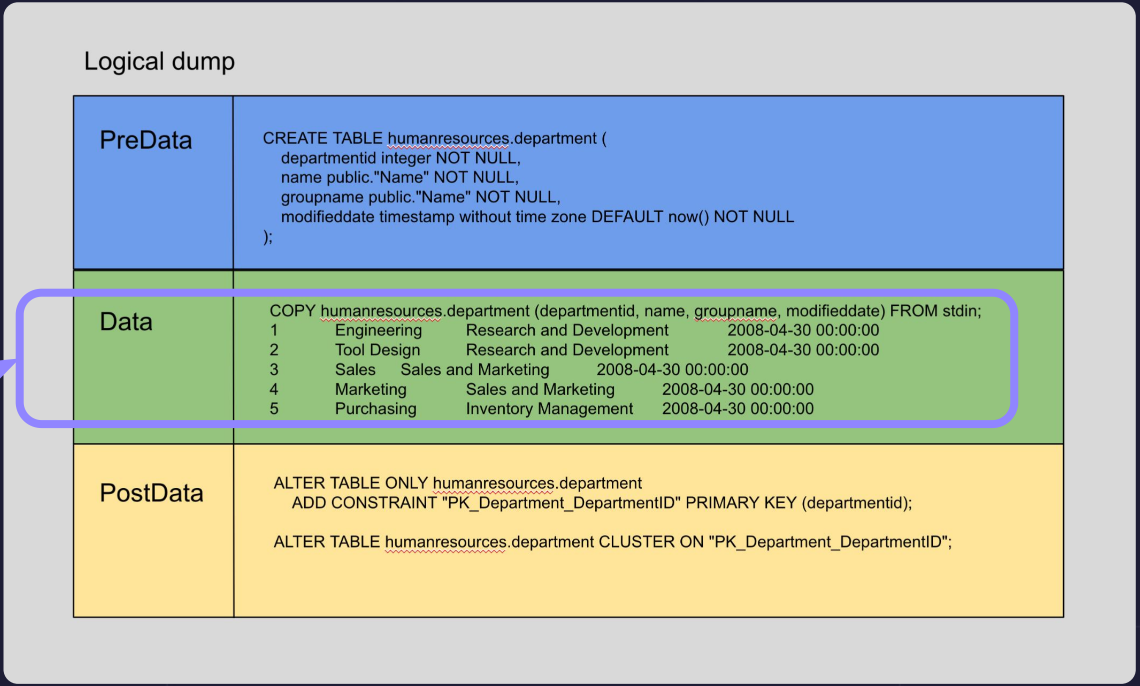
https://github.com/GreenmaskIO/greenmask

DUMP AND RESTORE CONCEPT

CORE CONCEPT:

- Delegate schema dumping to the vendor utilities
- Dump consist of 3 section
- Operate only with Data section

NEED TO CONTROL



Logical dump

PreData

```
CREATE TABLE humanresources.department (
    departmentid integer NOT NULL,
    name public."Name" NOT NULL,
    groupname public."Name" NOT NULL,
    modifieddate timestamp without time zone DEFAULT now() NOT NULL
);
```

Data

```
COPY humanresources.department (departmentid, name, groupname, modifieddate) FROM stdin;
1	Engineering	Research and Development	2008-04-30 00:00:00
2	Tool Design	Research and Development	2008-04-30 00:00:00
3	Sales	Sales and Marketing	2008-04-30 00:00:00
4	Marketing	Sales and Marketing	2008-04-30 00:00:00
5	Purchasing	Inventory Management	2008-04-30 00:00:00
```

PostData

```
ALTER TABLE ONLY humanresources.department
    ADD CONSTRAINT "PK_Department_DepartmentID" PRIMARY KEY (departmentid);

ALTER TABLE humanresources.department CLUSTER ON "PK_Department_DepartmentID";
```

TOC ARCHIVE LIBRARY

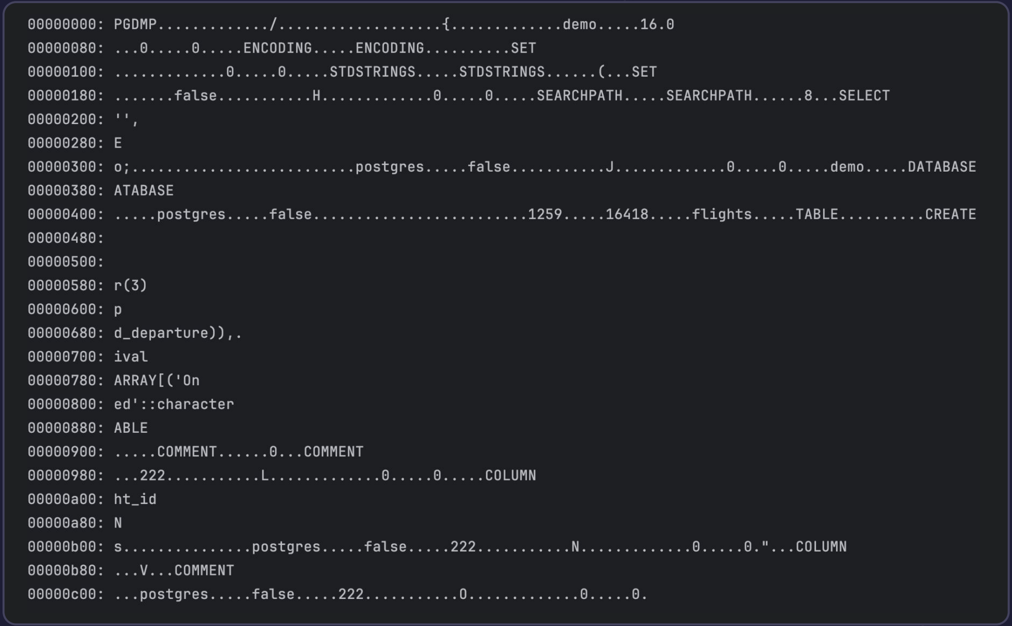
PURPOSE:

- Control dump and restore procedure
- Exclude data and objects when restoring
- Parallel execution
- Restoration dependencies resolving

CHARACTERISTICS:

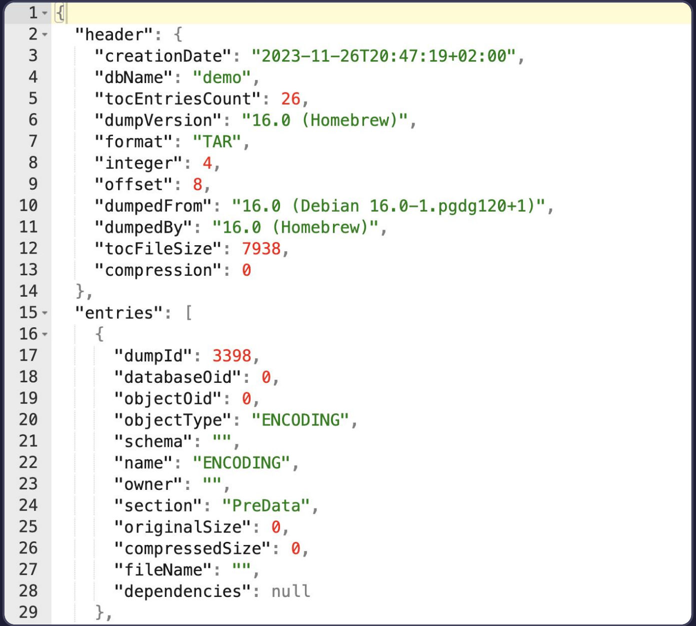
- Original library written in C
- Binary format of data
- Contains
  - Metadata
  - SQL commands
  - Data files references

TOC.DAT FILE EXAMPLE



```
00000000: PGDMP............./..................{............demo.....16.0
00000080: ...0.....0.....ENCODING.....ENCODING.........SET
00000100: ............0.....0.....STDSTRINGS.....STDSTRINGS......(...SET
00000180: .......false..........H............0.....0.....SEARCHPATH.....SEARCHPATH......8...SELECT
00000200: '',
00000280: E
00000300: o;.........................postgres.....false..........J............0.....0.....demo.....DATABASE
00000380: ATABASE
00000400: .....postgres.....false.........................1259.....16418.....flights.....TABLE.........CREATE
00000480:
00000500:
00000580: r(3)
00000600: p
00000680: d_departure)),.
00000700: ival
00000780: ARRAY[('On
00000800: ed'::character
00000880: ABLE
00000900: .....COMMENT......0...COMMENT
00000980: ...222..........L............0.....0.....COLUMN
00000a00: ht_id
00000a80: N
00000b00: s...............postgres.....false.....222..........N............0.....0."...COLUMN
00000b80: ...V...COMMENT
00000c00: ...postgres.....false.....222..........0............0.....0.
```

Binary metadata



```
{
  "header": {
    "creationDate": "2023-11-26T20:47:19+02:00",
    "dbName": "demo",
    "tocEntriesCount": 26,
    "dumpVersion": "16.0 (Homebrew)",
    "format": "TAR",
    "integer": 4,
    "offset": 8,
    "dumpedFrom": "16.0 (Debian 16.0-1.pgdg120+1)",
    "dumpedBy": "16.0 (Homebrew)",
    "tocFileSize": 7938,
    "compression": 0
  },
  "entries": [
    {
      "dumpId": 3398,
      "databaseOid": 0,
      "objectOid": 0,
      "objectType": "ENCODING",
      "schema": "",
      "name": "ENCODING",
      "owner": "",
      "section": "PreData",
      "originalSize": 0,
      "compressedSize": 0,
      "fileName": "",
      "dependencies": null
    },
```

Translated madata to JSON

COPY PARSER & NULL DETERMINATION

TABLE DEFINITION:

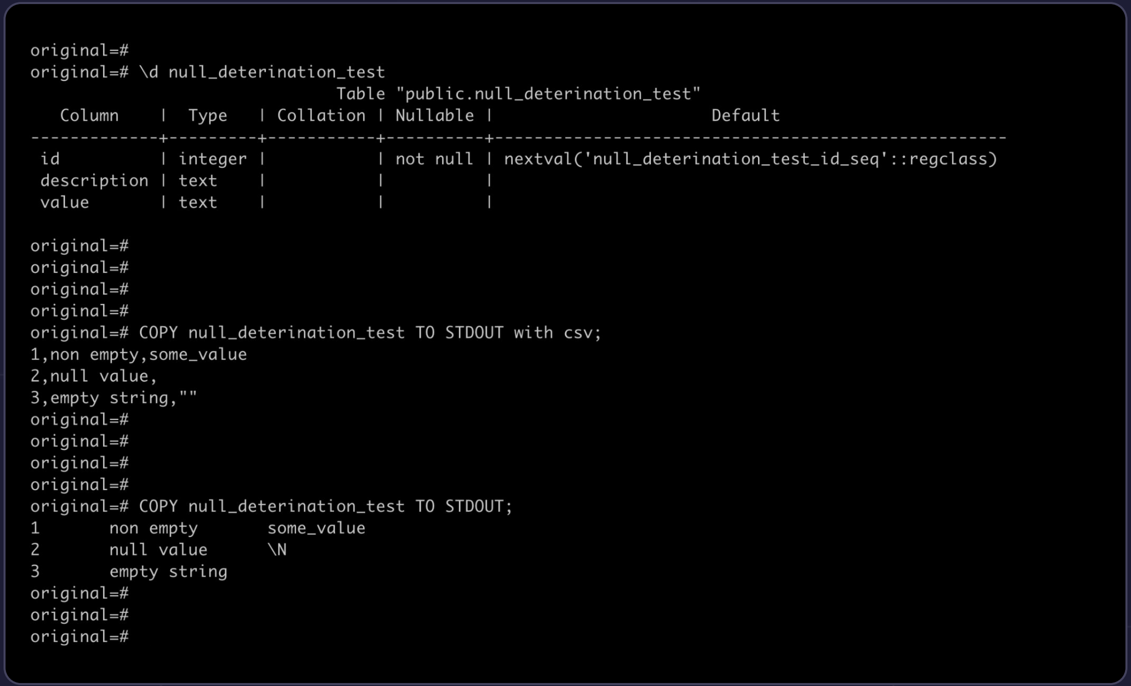
- Column value is nullable

COPY in CSV

- Has problem with NULL
- Empty = “”
- Null = empty string

COPY in TEXT

- Has strict NULL determination
- Collision is not expected



```
original=#
original=# \d null_deterination_test
                               Table "public.null_deterination_test"
   Column    |  Type   | Collation | Nullable |                      Default
-------------+---------+-----------+----------+----------------------------------------------------
 id          | integer |           | not null | nextval('null_deterination_test_id_seq'::regclass)
 description | text    |           |          |
 value       | text    |           |          |

original=#
original=#
original=#
original=#
original=# COPY null_deterination_test TO STDOUT with csv;
1,non empty,some_value
2,null value,
3,empty string,""
original=#
original=#
original=#
original=#
original=# COPY null_deterination_test TO STDOUT;
1       non empty       some_value
2       null value      \N
3       empty string
original=#
original=#
original=#
```

BACKWARD COMPATIBILITY

pg_dump and pg_restore delegation + TOC Archive Library ported + TEXT COPY format Ported = **Backward compatibility**

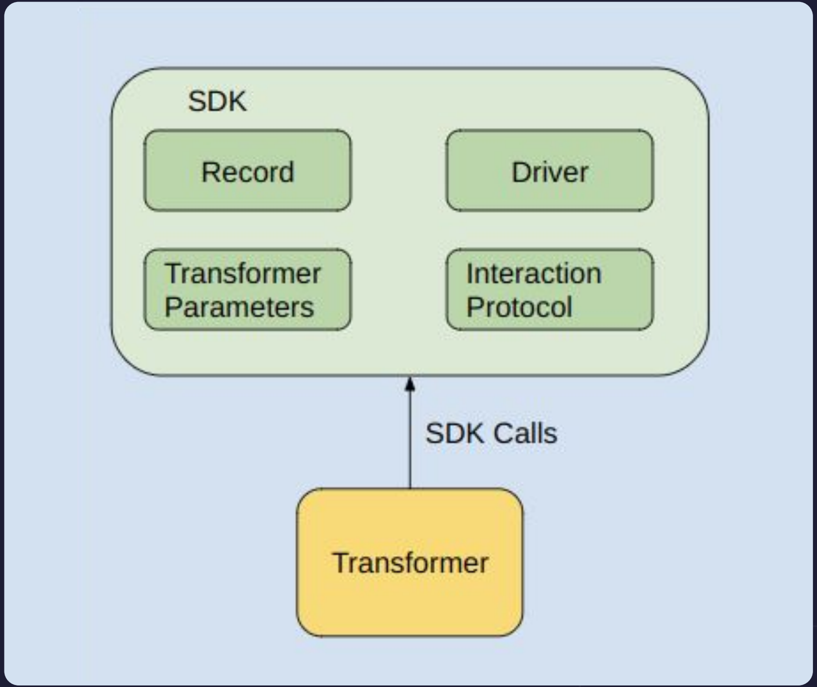
SDK AND EXTENSIBILITY

SUPPORTED LANGUAGES:

- Go
- Python (soon)

CHARACTERISTICS:

- Integrated with Database Driver
- Provides flexible transformer definition with parameters
- External transformer interaction protocol via PIPE



STORAGES

Directory

S3-Like storage (MinIO, AWS S3, etc.)

*

GREENMASK USAGE

BUILT-IN TRANSFORMERS

1. Cmd
2. Dict
3. Hash
4. Masking
5. NoiseDate
6. NoiseFloat
7. NoiseInt
8. RandomBool
9. RandomDate
10. RandomFloat
11. RandomInt
12. RandomString
13. RandomUuid
14. RandomLatitude
15. RandomLongitude
16. RandomUnixTime
17. RandomDayOfWeek
18. RandomDayOfMonth
19. RandomMonthName
20. RandomYearString
21. RandomCentury
22. RandomTimezone
23. RandomEmail
24. RandomUsername
25. RandomPassword
26. RandomMacAddress
27. RandomDomainName
28. RandomURL
29. RandomIPv4
30. RandomIPv6
31. RandomWord
32. RandomSentence
33. RandomParagraph
34. RandomCCType
35. RandomCCNumber
36. RandomCurrency
37. RandomAmountWithCurrency
38. RandomName
39. RandomLastName
40. RandomFirstName
41. RandomFirstNameMale
42. RealAddress
43. RandomFirstNameFemale
44. RandomTitleMale
45. RandomTitleFemale
46. RandomPhoneNumber
47. RandomTollFreePhoneNumber
48. RandomE164PhoneNumber
49. RealAddress
50. RegexpReplace
51. Replace
52. SetNull
53. Json
54. Template
55. TemplateRecord

https://greenmask.io/built_in_transformers/standard_transformers
https://greenmask.io/built_in_transformers/advanced_transformers

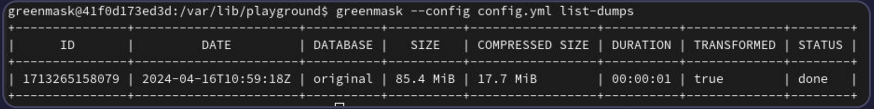
greenmask

VALIDATION

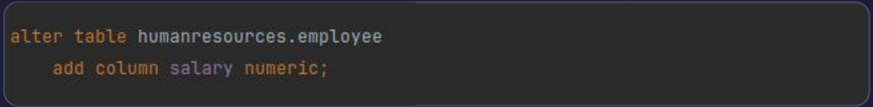
FEATURES:

- Control database schema diff
- Check config error
- Check DB constraint violations
- Perform transformation on the limited set
- Show the difference between original and transformed

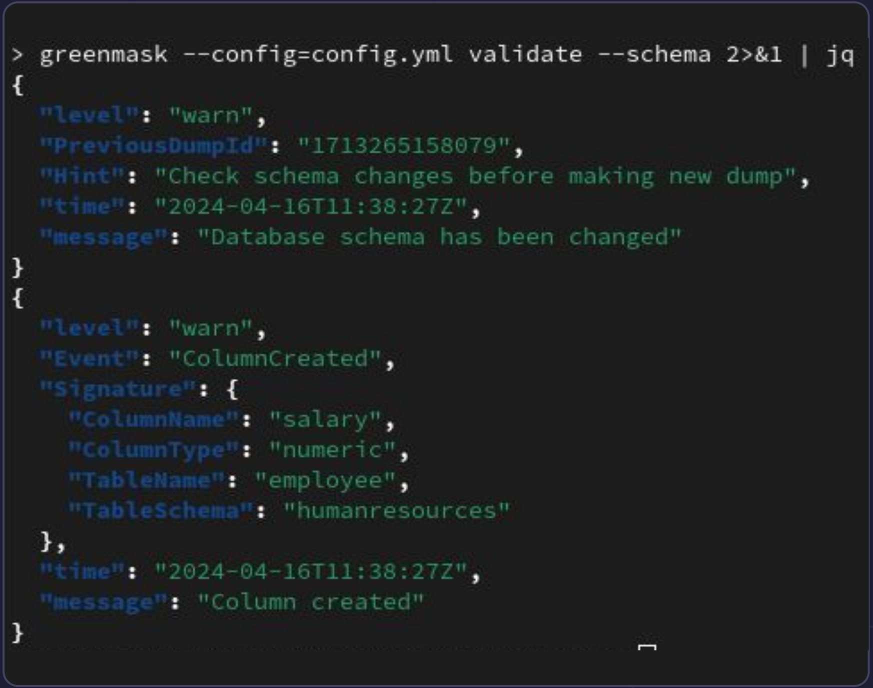
VALIDATE - SCHEMA DIFF



```
greenmask@41f0d173ed3d:/var/lib/playground$ greenmask --config config.yml list-dumps
+---------------+----------------------+----------+----------+-----------------+----------+-------------+--------+
|      ID       |         DATE         | DATABASE |   SIZE   | COMPRESSED SIZE | DURATION | TRANSFORMED | STATUS |
+---------------+----------------------+----------+----------+-----------------+----------+-------------+--------+
| 1713265158079 | 2024-04-16T10:59:18Z | original | 85.4 MiB | 17.7 MiB        | 00:00:01 | true        | done   |
+---------------+----------------------+----------+----------+-----------------+----------+-------------+--------+
```



```
alter table humanresources.employee
    add column salary numeric;
```

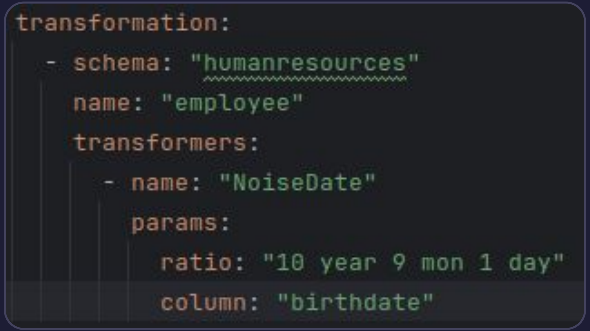


```
> greenmask --config=config.yml validate --schema 2>&1 | jq
{
  "level": "warn",
  "PreviousDumpId": "1713265158079",
  "Hint": "Check schema changes before making new dump",
  "time": "2024-04-16T11:38:27Z",
  "message": "Database schema has been changed"
}
{
  "level": "warn",
  "Event": "ColumnCreated",
  "Signature": {
    "ColumnName": "salary",
    "ColumnType": "numeric",
    "TableName": "employee",
    "TableSchema": "humanresources"
  },
  "time": "2024-04-16T11:38:27Z",
  "message": "Column created"
}
```

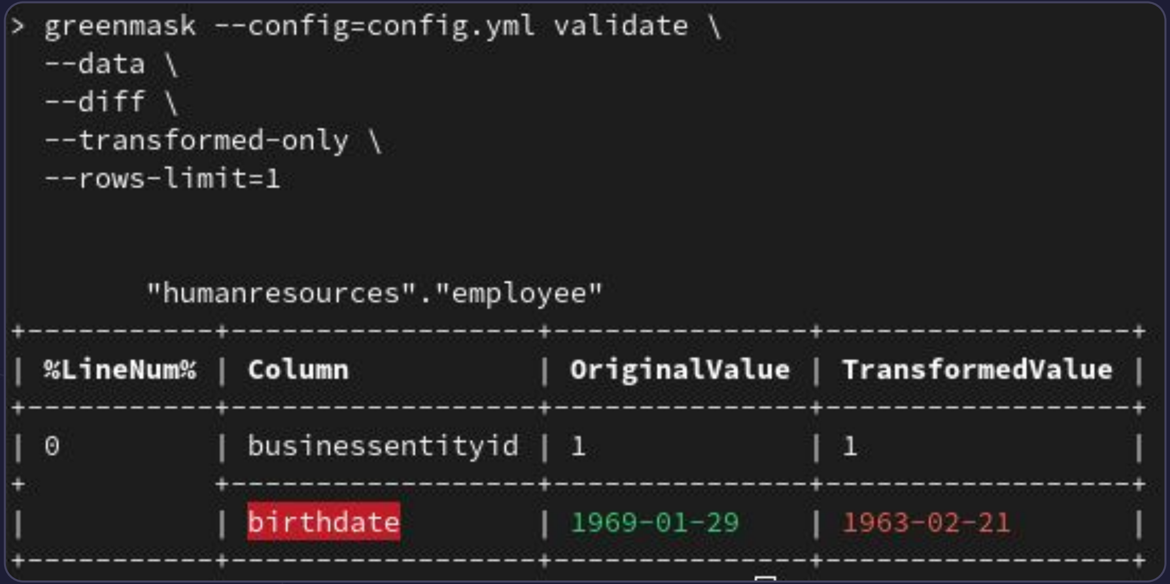
Non-zero exit if any changes

greenmask

VALIDATE - DATA DIFF

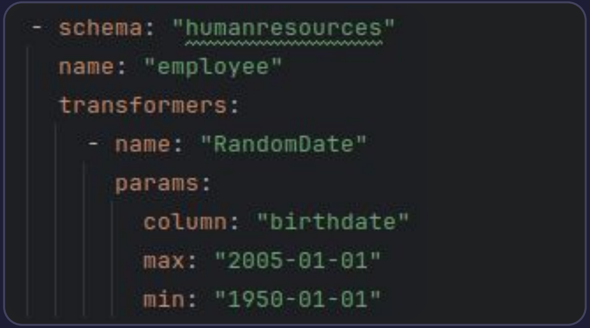


```yaml
transformation:
  - schema: "humanresources"
    name: "employee"
    transformers:
      - name: "NoiseDate"
        params:
          ratio: "10 year 9 mon 1 day"
          column: "birthdate"
```

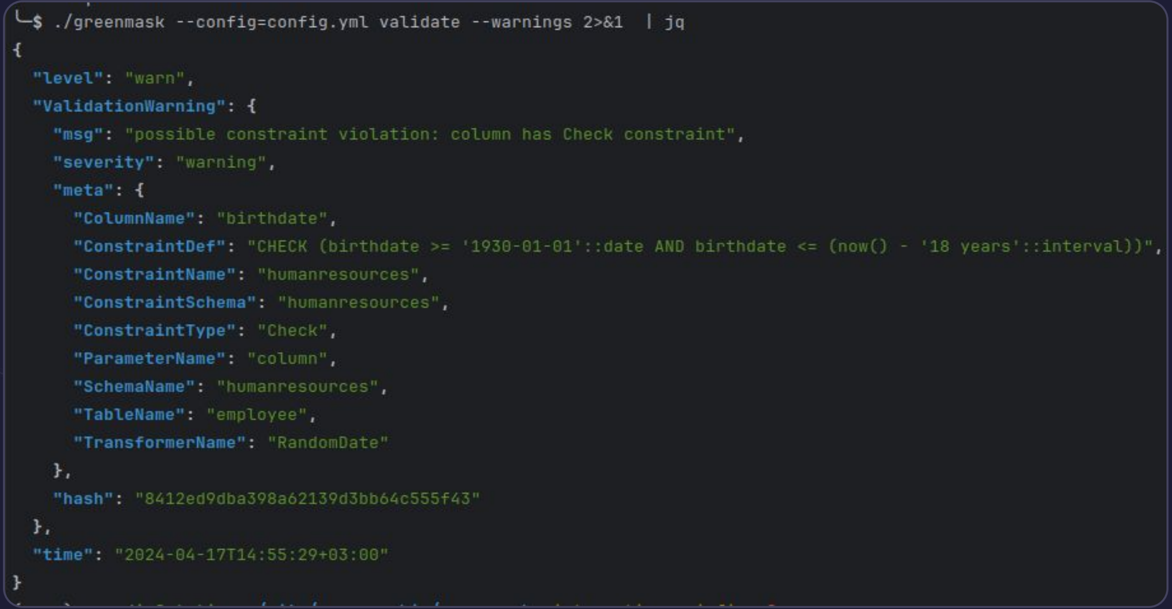


```
> greenmask --config=config.yml validate \
  --data \
  --diff \
  --transformed-only \
  --rows-limit=1


      "humanresources"."employee"
+-----------+------------------+----------------+------------------+
| %LineNum% | Column           | OriginalValue  | TransformedValue |
+-----------+------------------+----------------+------------------+
| 0         | businessentityid | 1              | 1                |
+           +------------------+----------------+------------------+
|           | birthdate        | 1969-01-29     | 1963-02-21       |
+-----------+------------------+----------------+------------------+
```

VALIDATE - WARNINGS

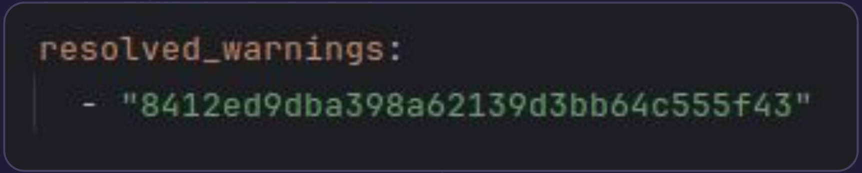


```yaml
- schema: "humanresources"
  name: "employee"
  transformers:
    - name: "RandomDate"
      params:
        column: "birthdate"
        max: "2005-01-01"
        min: "1950-01-01"
```



```
└$ ./greenmask --config=config.yml validate --warnings 2>&1  | jq
{
  "level": "warn",
  "ValidationWarning": {
    "msg": "possible constraint violation: column has Check constraint",
    "severity": "warning",
    "meta": {
      "ColumnName": "birthdate",
      "ConstraintDef": "CHECK (birthdate >= '1930-01-01'::date AND birthdate <= (now() - '18 years'::interval))",
      "ConstraintName": "humanresources",
      "ConstraintSchema": "humanresources",
      "ConstraintType": "Check",
      "ParameterName": "column",
      "SchemaName": "humanresources",
      "TableName": "employee",
      "TransformerName": "RandomDate"
    },
    "hash": "8412ed9dba398a62139d3bb64c555f43"
  },
  "time": "2024-04-17T14:55:29+03:00"
}
```

VALIDATE - WARNINGS RESOLVING



```
resolved_warnings:
  - "8412ed9dba398a62139d3bb64c555f43"
```

You can resolve warning by adding it hash to resolved warnings list

greenmask

CUSTOM LOGIC - TEMPLATE RECORD

```
- name: "TemplateRecord"
  params:
    columns:
      - "created_at"
      - "updated_at"
    template: >
      {{ $val := .GetColumnValue "created_at" }}
      {{ if isNotNull $val }}
          {{ $createdAtValue := now }}
          {{ $maxUpdatedDate := date_modify "24h" $createdAtValue }}
          {{ $updatedAtValue := randomDate $createdAtValue $maxUpdatedDate }}
          {{ .SetColumnValue "created_at" $createdAtValue }}
          {{ .SetColumnValue "updated_at" $updatedAtValue }}
      {{ end }}
    validate: true
```

column name	original value	transformed
created_at	2021-01-20 07:01:00.513325+00	2023-12-17 19:37:29.910054Z
updated_at	2021-08-09 21:27:00.513325+00	2023-12-18 10:05:25.828498Z

EXAMPLE REQUIREMENTS:

- Resolve functional dependencies
- Generate new created_at value
- Generate random updated_at value start from created_at

OTHER FEATURES

- Works with huge databases (tested with 1TB)
- Custom DB types registering in driver
- Partitioned tables inheritance
- Declarative
- Easy integrated into your CI/CD system
- Parallel execution
- Cross platform
- Section scripts (pre-data, data, post-data)
- Large objects dumping

*

04

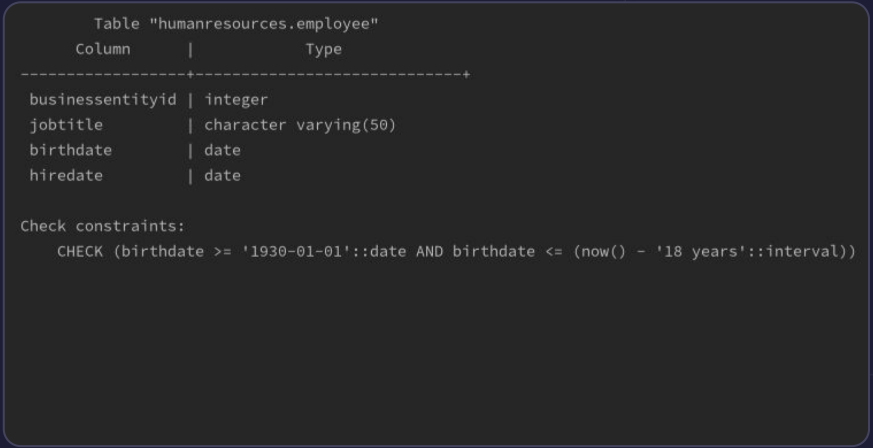
GREENMASK TEASED FEATURE

DYNAMIC PARAMETERS

- Getting parameter value from the record
- Resolving functional dependencies
- Resolve constraint violations
- Keeping database consistent

*Coming soon (May)

DYNAMIC PARAMETERS



```
        Table "humanresources.employee"
      Column      |             Type
------------------+----------------------------+
 businessentityid | integer
 jobtitle         | character varying(50)
 birthdate        | date
 hiredate         | date

Check constraints:
    CHECK (birthdate >= '1930-01-01'::date AND birthdate <= (now() - '18 years'::interval))
```

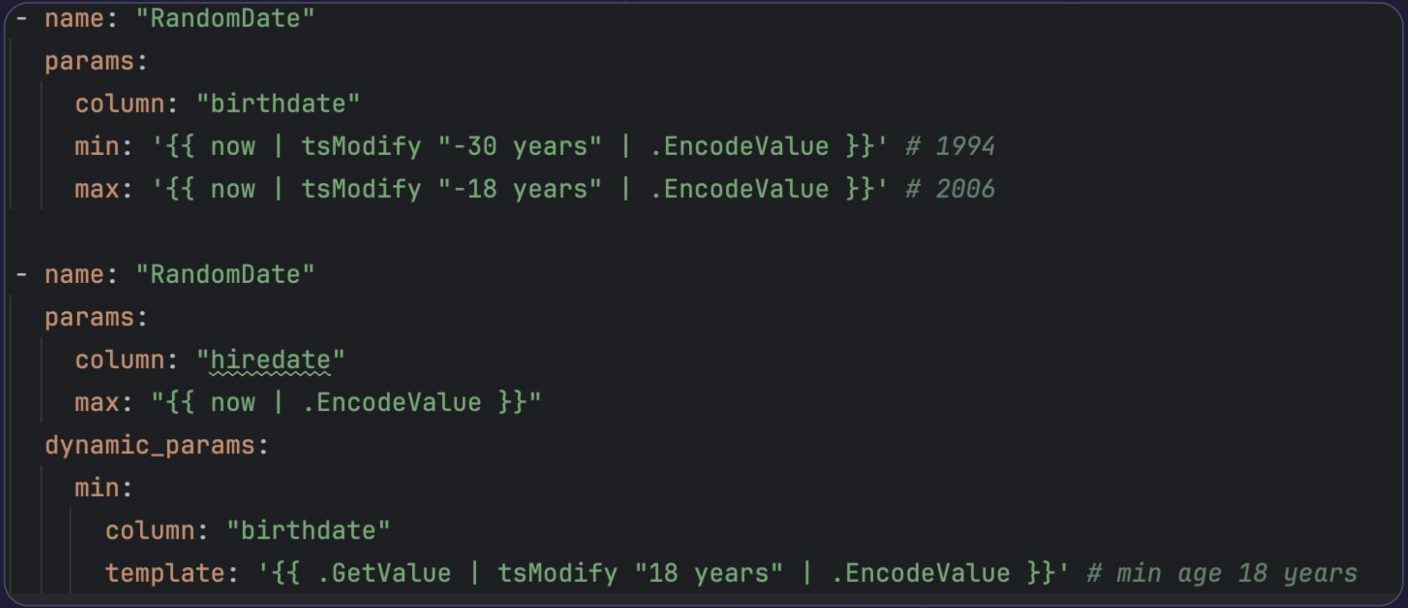
REQUIREMENTS:

- hiredate > birthdate (Functional dependency)
- now() - hiredate >= 18 years (Constraint)

*Coming soon (May)

greenmask

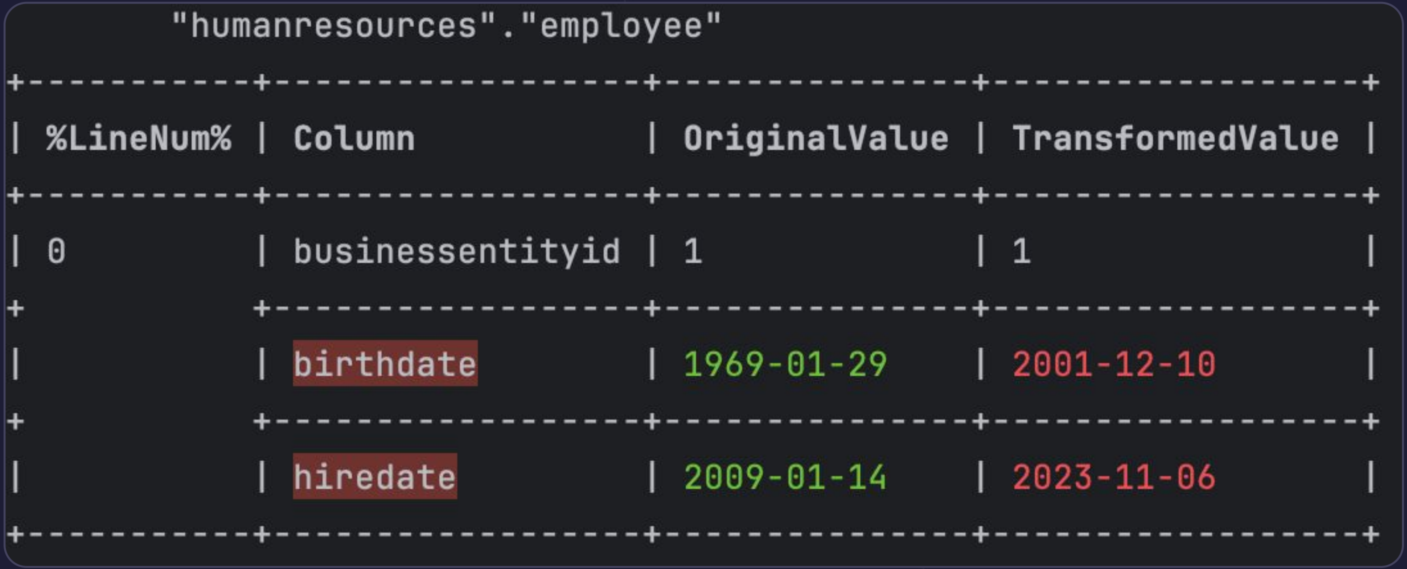
DYNAMIC PARAMETERS — CONFIG



```yaml
- name: "RandomDate"
  params:
    column: "birthdate"
    min: '{{ now | tsModify "-30 years" | .EncodeValue }}' # 1994
    max: '{{ now | tsModify "-18 years" | .EncodeValue }}' # 2006

- name: "RandomDate"
  params:
    column: "hiredate"
    max: "{{ now | .EncodeValue }}"
  dynamic_params:
    min:
      column: "birthdate"
      template: '{{ .GetValue | tsModify "18 years" | .EncodeValue }}' # min age 18 years
```

*Coming soon (May)

DYNAMIC PARAMETERS — RESULT



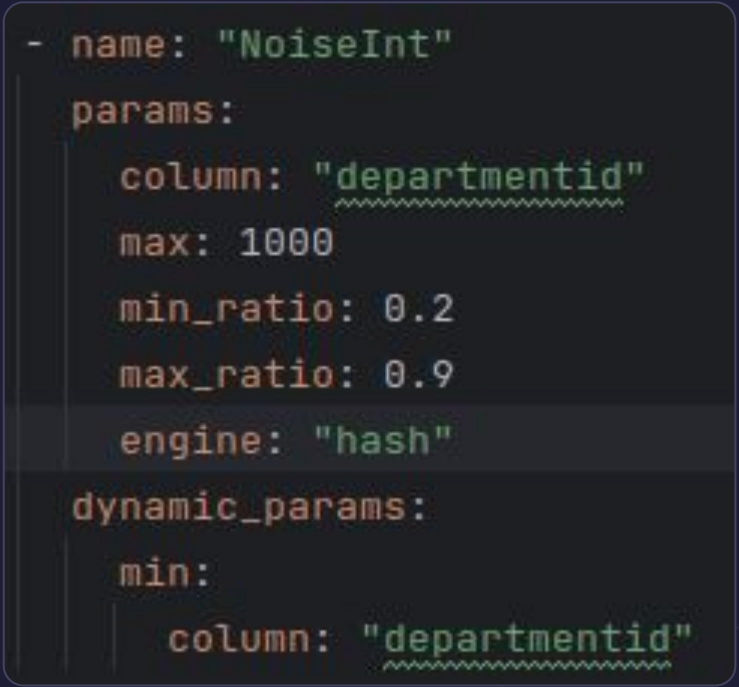
```
        "humanresources"."employee"
+-----------+------------------+---------------+-------------------+
| %LineNum% | Column           | OriginalValue | TransformedValue |
+-----------+------------------+---------------+-------------------+
| 0         | businessentityid | 1             | 1                 |
+           +------------------+---------------+-------------------+
|           | birthdate        | 1969-01-29    | 2001-12-10        |
+           +------------------+---------------+-------------------+
|           | hiredate         | 2009-01-14    | 2023-11-06        |
+-----------+------------------+---------------+-------------------+
```

*Coming soon (May)

DETERMINISTIC TRANSFORMATIONS

FEATURES:

- Cryptographic functions
  - Sha2
  - Sha3
- Unified hash function
  - SipHash



```yaml
- name: "NoiseInt"
  params:
    column: "departmentid"
    max: 1000
    min_ratio: 0.2
    max_ratio: 0.9
    engine: "hash"
  dynamic_params:
    min:
      column: "departmentid"
```

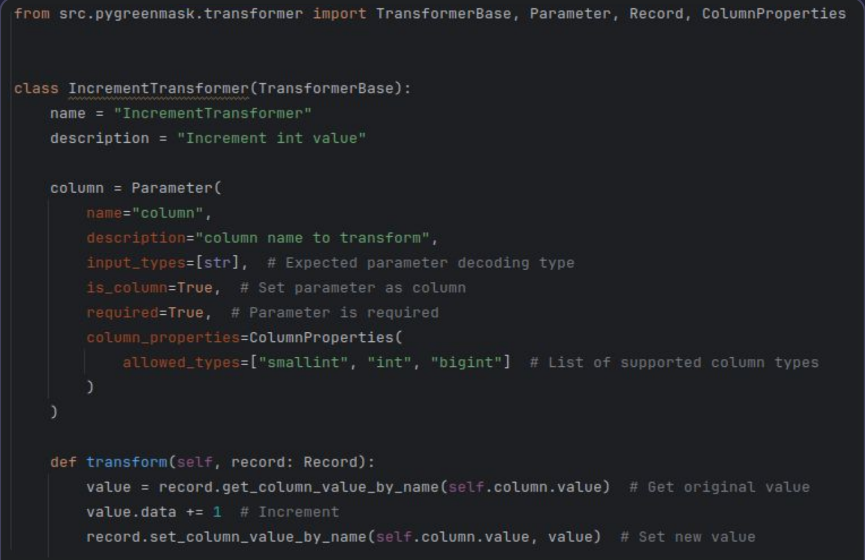
*Coming soon (May)

EXTERNAL TRANSFORMER EXAMPLE

FEATURES:

- Simple definition
- Multicolumn support
- Focus only on transformation logic
- Autodiscovery by core utility
- Can be integrated with your framework and any service logic

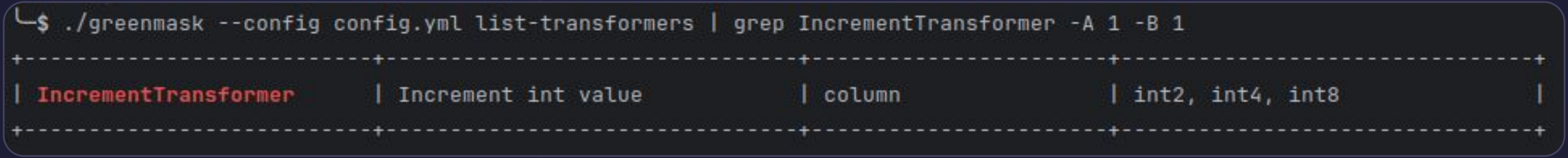
*Coming soon (May-June)



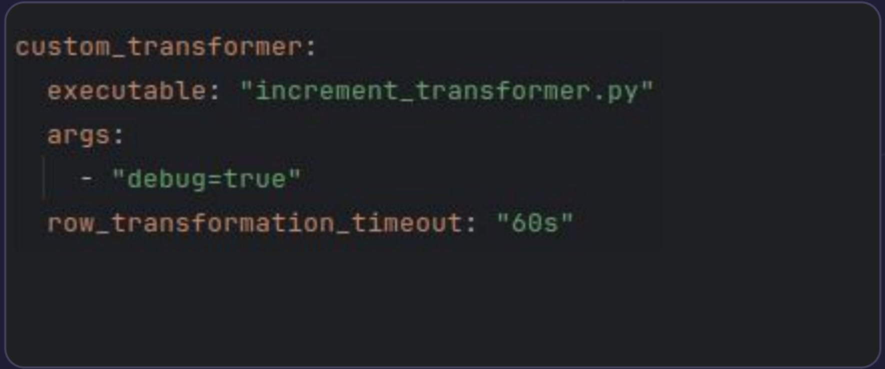
```python
from src.pygreenmask.transformer import TransformerBase, Parameter, Record, ColumnProperties


class IncrementTransformer(TransformerBase):
    name = "IncrementTransformer"
    description = "Increment int value"

    column = Parameter(
        name="column",
        description="column name to transform",
        input_types=[str],  # Expected parameter decoding type
        is_column=True,  # Set parameter as column
        required=True,  # Parameter is required
        column_properties=ColumnProperties(
            allowed_types=["smallint", "int", "bigint"]  # List of supported column types
        )
    )

    def transform(self, record: Record):
        value = record.get_column_value_by_name(self.column.value)  # Get original value
        value.data += 1  # Increment
        record.set_column_value_by_name(self.column.value, value)  # Set new value
```

Teased Python library (coming soon)

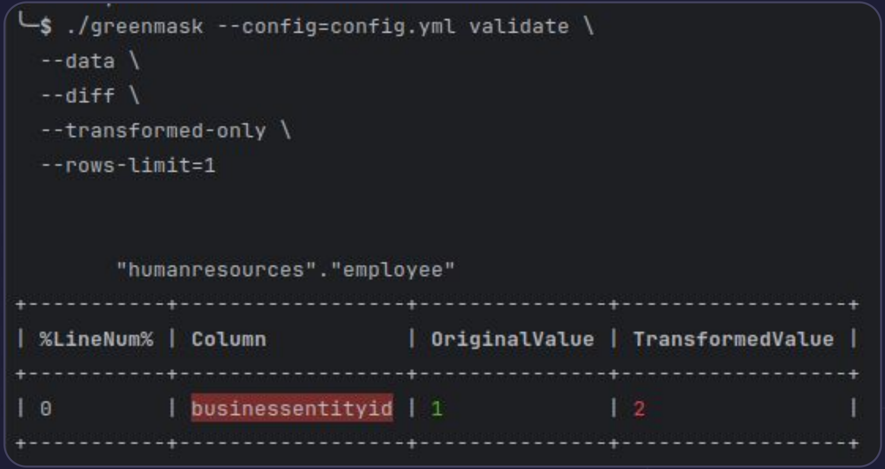
EXTERNAL TRANSFORMER AUTO-DISCOVERY



```
└$ ./greenmask --config config.yml list-transformers | grep IncrementTransformer -A 1 -B 1
+---------------------------+---------------------------------+-----------------------+---------------------------------+
| IncrementTransformer      | Increment int value             | column                | int2, int4, int8                |
+---------------------------+---------------------------------+-----------------------+---------------------------------+
```



```
custom_transformer:
  executable: "increment_transformer.py"
  args:
    - "debug=true"
  row_transformation_timeout: "60s"
```



```
└$ ./greenmask --config=config.yml validate \
  --data \
  --diff \
  --transformed-only \
  --rows-limit=1


      "humanresources"."employee"
+-----------+----------------+-------------+------------------+
| %LineNum% | Column         | OriginalValue | TransformedValue |
+-----------+----------------+-------------+------------------+
| 0         | businessentityid | 1           | 2                |
+-----------+----------------+-------------+------------------+
```

*Coming soon (May-June)

*

PLANS & ROAD MAP

OUR GOALS

- Help businesses to improve their services
- Make data safe and private
- Develop Open-Source Test Data Management Platform
- Implement reusable and extensible code base (Not only for DBMS)

COMING SOON (APRIL-JUNE)

- Deterministic library and transformers
- Dynamic parameters
- Database subset
- Auto PK/FK transformations
- Transformation conditions
- New built-in transformers
- Python library (SDK)

@greenmask_community

ROADMAP & CONTRIBUTION

- Unique transformations
- Transformation patterns
- AI Assistant & External integrations
- Transformation patterns
- New DBMS support (OLAP, OLTP)
- Web interface
- Scalable service

@greenmask_community

*

THANK YOU!