

被ばく線量シミュレーションシステム データベース準備マニュアル

2022年6月

国立研究開発法人日本原子力研究開発機構

目次

1. はじめに	1
2. 動作環境	1
3. 構成ファイル	1
4. 前提ソフトウェア	2
5. セットアップ手順	3
5.1 ユーザの作成	3
5.2 資材の配置	3
5.3 前提ソフトウェアのインストールおよびセットアップ	3
5.3.1 PostgreSQLのインストールおよびセットアップ	3
5.3.2 Pythonのインストール	5
5.3.3 関連パッケージのインストール	6
5.4 データベースの構築	6
5.4.1 データベース環境構築	6
5.4.2 仮想環境の作成、Pythonライブラリのインストール、テーブル作成の実施	7
5.4.3 テーブル、シーケンスの初期化	8
5.5 データベースへのデータの登録	8
5.5.1 統合マップ、バックグラウンド、移動手段のマスタデータ登録	8
5.5.2 統合マップ、バックグラウンドのデータ作成・登録	9
5.5.3 駅・経路のデータ作成・登録	10
5.5.4 駅・経路のデータ更新	11
5.5.5 データのエクスポート	13

1. はじめに

本書では、被ばく線量シミュレーションシステムのデータベースを構築する手順について記載します。

2. 動作環境

被ばく線量シミュレーションシステムのデータベース準備用サーバの動作環境を表 2-1に示します。

表 2-1 動作環境

OS	Ubuntu 20.04LTS
推奨メモリサイズ	8GB以上
推奨ディスク容量	100GB以上

3. 構成ファイル

被ばく線量シミュレーションシステムのデータベース準備のために必要な構成ファイルを表 3-1に示します。

表 3-1 構成ファイル

ファイル名	ファイル内容
SEED.zip	被ばく線量シミュレーションシステムインストールデータ
out_data.txt	統合マップのマスターデータ (50mメッシュ) ※本手順書では2021年度に作成された6,005,733件のデータを想定して記載している
m_background.csv	バックグラウンドのマスターデータ (50mメッシュ) ※本手順書では2021年度に作成された5,437,621件のデータを想定して記載している
m_transporttype.csv	移動手段のマスターデータ
N02-20_GML.zip	国土数値情報 (https://nlftp.mlit.go.jp/ksj/gml/datalist/KsjTmplt-N02-v2_3.html) からダウンロードした鉄道情報 ※本手順書では令和2年度データ (https://nlftp.mlit.go.jp/ksj/gml/data/N02/N02-20/N02-20_GML.zip) を用いている
station_batch.zip	鉄道における駅・経路情報を登録するためのスクリプト
train_move_time.csv	常磐線 (新地～勿来) の駅間の時刻のデータ

4. 前提ソフトウェア

被ばく線量シミュレーションシステムのデータベース準備のための前提ソフトウェアを表 4-1に示します。

表 4-1 前提ソフトウェア

データベース	PostgreSQL 13
GIS 拡張	PostGIS 3.1
	Pgrouting 2.6
Python	Python 3.8

5. セットアップ手順

本章では、被ばく線量シミュレーションシステムのデータベース準備のための手順を記載します。#で開始されるコマンドはrootユーザで行ってください。また、<>内の項目は実際の項目へ読み替えてください。

5.1 ユーザの作成

以下のコマンドを実行し、seedユーザを作成します。

○seedユーザ作成

```
# adduser -q --gecos "" --disabled-login seed
```

○seedユーザ作成確認

```
# id seed
```

→seedユーザが作成されていることを確認する。

○資材配置用ディレクトリ作成

```
# su - seed
```

```
$ mkdir -pv ~/work/db_setup/module
```

```
$ ls -ld ~/work/db_setup/module
```

→ディレクトリが作成されていることを確認する。

5.2 資材の配置

5.1で作成した資材配置用ディレクトリへ以下の通り資材を配置します。

- ・ SEED.zip : /home/seed/work/db_setup/module配下
- ・ station_batch.zip : /home/seed/work/db_setup配下
- ・ N02-20_GML.zip : /home/seed/work/db_setup配下

5.3 前提ソフトウェアのインストールおよびセットアップ

本節の手順は、対象サーバがインターネットへ接続された状態で行ってください。

5.3.1 PostgreSQLのインストールおよびセットアップ

5.3.1.1 PostgreSQL13のインストール準備

以下のコマンドを実行し、PostgreSQLのインストール準備を実施します。

○リポジトリ追加

```
# sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

```
# ls -l /etc/apt/sources.list.d/pgdg.list
```

→0バイト以上のファイルが存在することを確認する。

○PostgreSQLの認証キーの追加

```
# wget https://www.postgresql.org/media/keys/ACCC4CF8.asc -P /tmp/
# ls -l /tmp/ACCC4CF8.asc
→0バイト以上のファイルが存在することを確認する。
# apt-key add /tmp/ACCC4CF8.asc
→「OK」が表示されることを確認する。
```

5.3.1.2 PostgreSQL13のインストール

以下のコマンドを実行し、PostgreSQL13のインストールを実施します。

○パッケージ更新

```
# apt update
```

○PostgreSQL13およびGISモジュールのインストール

```
# apt install postgresql-13 postgresql-13-postgis-3 postgresql-13-pgrouting
→「Do you want to continue? [Y/n]」が表示されたら「Y」を入力後にEnterを押下して継続する。
```

○インストール確認

```
# dpkg -l | grep postgresql-13
→結果が表示されることを確認する。
```

5.3.1.3 PostgreSQLのセットアップ

以下のコマンドを実行し、PostgreSQL13のセットアップを実施します。

○/etc/postgresql/13/main/pg_hba.confの設定

```
# cp -av /etc/postgresql/13/main/pg_hba.conf{,.org}
# vi /etc/postgresql/13/main/pg_hba.conf
→以下の通り設定する。
<88行目付近>
# Database administrative login by Unix domain socket
#local all postgres peer ←コメントアウト
<93行目付近>#
# "local" is for Unix domain socket connections only
local all all password ←「peer」から変更
# IPv4 local connections:
host all all 127.0.0.1/32 password ←「md5」から変更
# IPv6 local connections:
#host all all ::1/128 md5 ←コメントアウトする
# Allow replication connections from localhost, by a user with the
```

```
# replication privilege.
#local replication all peer ←コメントアウト
host replication all 127.0.0.1/32 password ←「md5」から変更
#host replication all ::1/128 md5 ←コメントアウト

○/etc/postgresql/13/main/pg_hba.confの設定内容確認
# diff /etc/postgresql/13/main/pg_hba.conf{,.org}
→変更した箇所しか差分がないことを確認する。
```

5.3.1.4 postgresqlユーザへのパスワード設定

以下のコマンドを実行し、postgresqlユーザへのパスワード設定を実施します。

```
○PostgreSQLへのログイン
# su - postgres
$ psql
→ログインに成功し、プロンプトが「postgres=#」へ変更されることを確認する。

○postgresqlユーザのパスワード設定
postgres=# ALTER USER "postgres" WITH PASSWORD '<パスワード>';
→「ALTER ROLE」が表示されることを確認する。
postgres=# quit;
$ exit

○PostgreSQLの再起動
# systemctl restart postgresql
# systemctl status postgresql
→postgresqlが起動していることを確認する。
```

5.3.2 Pythonのインストール

以下のコマンドを実行し、Python3のインストールを実施します。

```
○パッケージ更新
# apt update

○Pythonインストール
# apt install python3 python3-dev python3-venv
→「Do you want to continue? [Y/n]」が表示されたら「Y」を入力後にEnterを押下して継続する。
```

○インストール確認

```
# dpkg -l | grep python3
```

→結果が表示されることを確認する。

5.3.3 関連パッケージのインストール

以下のコマンドを実行し、本システムに必要なパッケージのインストールを実施します。

○パッケージ更新

```
# apt update
```

○関連パッケージのインストール

```
# apt install gcc gdal-bin libgl1-mesa-dev libpq-dev unzip postgis
```

→「Do you want to continue? [Y/n]」が表示されたら「Y」を入力後にEnterを押下して継続する。

○インストール確認

```
# dpkg -l | grep "gcc¥|gdal-bin¥|libgl1-mesa-dev¥|libpq-dev¥|unzip¥|postgis"
```

→結果が表示されることを確認する。

5.4 データベースの構築

5.4.1 データベース環境構築

以下のコマンドを実行し、本システム用のデータベース環境を構築します。

○seed_usrの作成

```
# su - postgres
```

```
$ createuser seed_usr
```

→postgresユーザのパスワードを入力し、エラーとならずに処理が終了することを確認する。

```
$ psql
```

```
postgres=# ¥du
```

→「seed_usr」が存在していることを確認

○seed_usrユーザのパスワード設定および権限付与

```
postgres=# ALTER USER "seed_usr" WITH PASSWORD '< パスワード >';
```

→「ALTER ROLE」が表示されることを確認する。

```
postgres=# ALTER USER "seed_usr" WITH SUPERUSER;
```

→「ALTER ROLE」が表示されることを確認する。

```
postgres=# quit;
```

```
$ exit
```

○データベース作成

```
# createdb -E UTF8 -O seed_usr -U seed_usr seed_db
```

→seed_usrユーザのパスワードを入力し、エラーとならずに処理が終了することを確認する。

○スキーマ作成、拡張GISの有効化、パスの設定

```
# psql -U seed_usr -d seed_db
```

→seed_usrユーザのパスワードを入力し、ログインする

```
seed_db=# CREATE SCHEMA seed_schema;
```

→「CREATE SCHEMA」が表示されることを確認する。

```
seed_db=# CREATE EXTENSION IF NOT EXISTS "postgis" SCHEMA seed_schema;
```

→「CREATE EXTENSION」が表示されることを確認する。

```
seed_db=# CREATE EXTENSION IF NOT EXISTS "pgrouting" SCHEMA seed_schema;
```

→「CREATE EXTENSION」が表示されることを確認する。

```
seed_db=# ALTER USER seed_usr SET SEARCH_PATH TO seed_schema;
```

→「ALTER ROLE」が表示されることを確認する。

```
seed_db=# quit;
```

5.4.2 仮想環境の作成、Pythonライブラリのインストール、テーブル作成の実施

以下のコマンドを実行し、本システム用のPythonの仮想環境を作成し、Pythonライブラリのインストール、およびデータベースのテーブル作成を実施します。

○仮想環境の作成

```
# su - seed
```

```
$ cd ~/work/db_setup/module; ls -l SEED.zip
```

→ファイルが存在することを確認する。

```
$ unzip SEED.zip
```

→エラーとならずに処理が終了することを確認する。

```
$ cd SEED
```

```
$ python3 -m venv env
```

→エラーとならずに処理が終了することを確認する。

```
$ source env/bin/activate
```

→プロンプトに「(env)」が付与されて表示されることを確認する。

○Pythonライブラリのインストール

```
$ pip install -r requirements.txt
```

→末尾に「Successfully installed」が表示されることを確認する。

○テーブル作成

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
→エラーとならずに処理が終了することを確認する。
$ exit
```

5.4.3 テーブル、シーケンスの初期化

以下のコマンドを実行し、本システム用のテーブルおよびシーケンスの初期化を実施します。

※本手順は初回構築時には必要ありません。データ入替時等を実施してください。

○テーブルの初期化

```
# psql -U seed_usr -d seed_db
→seed_usrユーザのパスワードを入力し、ログインする。
seed_db=# TRUNCATE m_transporttype;
seed_db=# TRUNCATE m_background;
seed_db=# TRUNCATE doserate_data_master;
seed_db=# TRUNCATE doserate_data;
seed_db=# TRUNCATE railroad_section_work;
seed_db=# TRUNCATE station;
seed_db=# TRUNCATE railroad_section;
seed_db=# TRUNCATE rail_node;
→エラーとならずに処理が終了することを確認する。
```

○シーケンスの初期化

```
seed_db=# SELECT SETVAL ('m_transporttype_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('m_background_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('doserate_data_master_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('doserate_data_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('railroad_section_work_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('station_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('railroad_section_id_seq', 1, FALSE);
seed_db=# SELECT SETVAL ('rail_node_id_seq', 1, FALSE);
→エラーとならずに処理が終了することを確認する。
seed_db=# quit;
```

5.5 データベースへのデータの登録

5.5.1 統合マップ、バックグラウンド、移動手段のマスタデータ登録

以下のコマンドを実行し、統合マップ、バックグラウンド、移動手段のマスタデータを登録します。

○マスタデータの登録

```
# psql -U seed_usr -d seed_db
```

→seed_usrユーザのパスワードを入力し、ログインする

```
seed_db=# COPY m_background(m_bg_citycd, m_bg_bgvalue, m_bg_longitude, m_bg_latitude)
FROM '/home/seed/work/db_setup/m_background.csv' WITH CSV DELIMITER ',' header;
```

→「COPY 5437621」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# COPY doserate_data_master(ddm_number,ddm_mesh,ddm_bif_hensa,ddm_aft_hensa,
ddm_bif,ddm_aft,ddm_y_zahyo,ddm_x_zahyo,ddm_genpatu) FROM '/home/seed/work/db_setu
p/out_data.txt' WITH CSV DELIMITER ' ';
```

→「COPY 6005733」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# COPY m_transporttype(m_tr_code,m_tr_typename,m_tr_pathname,m_tr_coefficient)
FROM '/home/seed/work/db_setup/m_transporttype.csv' WITH CSV DELIMITER ',' header;
```

→「COPY 5」が表示され、エラーとならずに処理が終了することを確認する。

5.5.2 統合マップ、バックグラウンドのデータ作成・登録

以下のコマンドを実行し、統合マップ、バックグラウンドのデータの作成および登録を実施します。

○統合マップの原点座標からの移動、メッシュの頂点座標を挿入

```
seed_db=# UPDATE doserate_data_master SET ddm_x_zahyo_ne = ddm_x_zahyo + 328351
+ 25, ddm_y_zahyo_ne = ddm_y_zahyo + 4061650 + 25, ddm_x_zahyo_nw = ddm_x_zahyo +
328351 - 25, ddm_y_zahyo_nw = ddm_y_zahyo + 4061650 + 25, ddm_x_zahyo_sw = ddm_x
_zahyo + 328351 - 25, ddm_y_zahyo_sw = ddm_y_zahyo + 4061650 - 25, ddm_x_zahyo_se =
ddm_x_zahyo + 328351 + 25, ddm_y_zahyo_se = ddm_y_zahyo + 4061650 - 25;
```

→「UPDATE 6005733」が表示され、エラーとならずに処理が終了することを確認する。

○統合マップの座標系変換、メッシュ作成

```
seed_db=# INSERT INTO doserate_data(dd_ym, dd_code, dd_lat, dd_lng, dd_geometry_point,
dd_geometry_mesh, dd_dose) SELECT 2021, ddm_number, ST_Y(ST_Transform(ST_SetSRI
D(ST_MakePoint(ddm_x_zahyo + 328351, ddm_y_zahyo + 4061650), 32654), 4326)), ST_X(ST
_Transform(ST_SetSRID(ST_MakePoint(ddm_x_zahyo + 328351, ddm_y_zahyo + 4061650), 3
2654), 4326)), ST_Transform(ST_SetSRID(ST_MakePoint(ddm_x_zahyo + 328351, ddm_y_zah
yo + 4061650), 32654), 4326), ST_MakePolygon(ST_Transform(ST_MakeLine( ARRAY[ ST_Se
tSRID(ST_MakePoint(ddm_x_zahyo_ne,ddm_y_zahyo_ne),32654), ST_SetSRID(ST_MakePoint(
ddm_x_zahyo_nw,ddm_y_zahyo_nw),32654), ST_SetSRID(ST_MakePoint(ddm_x_zahyo_sw,ddm
_y_zahyo_sw),32654), ST_SetSRID(ST_MakePoint(ddm_x_zahyo_se,ddm_y_zahyo_se),32654), S
T_SetSRID(ST_MakePoint(ddm_x_zahyo_ne,ddm_y_zahyo_ne),32654)]), 4326)), power(10, ddm
_aft) FROM doserate_data_master;
```

→「INSERT 0 6005733」が表示され、エラーとならずに処理が終了することを確認する。

○メッシュと中心点へインデックスの付与

```
seed_db=# CREATE INDEX doserate_data_dd_geometry_mesh_idx ON doserate_data USING
GIST (dd_geometry_mesh);
```

→「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX doserate_data_dd_geometry_point_idx ON doserate_data USING
GIST (dd_geometry_point);
```

→「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

○バックグラウンドのポイントのジオメトリ化

```
seed_db=# UPDATE m_background SET m_bg_geomerty = ST_SetSRID(ST_MakePoint(m_bg
_longitude, m_bg_latitude),4326);
```

→「UPDATE 5437621」が表示され、エラーとならずに処理が終了することを確認する。

○バックグラウンドのポイントへインデックスの付与

```
seed_db=# CREATE INDEX m_background_m_bg_geomerty_idx ON m_background USING
GIST (m_bg_geomerty);
```

→「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

○統合マップへバックグラウンド値の挿入

```
seed_db=# UPDATE doserate_data SET dd_bgvalue = m_bg_bgvalue FROM m_background
WHERE ST_Intersects(dd_geometry_mesh,m_bg_geomerty);
```

→「UPDATE 5432715」が表示され、エラーとならずに処理が終了することを確認する。

5.5.3 駅・経路のデータ作成・登録

以下のコマンドを実行し、鉄道における駅・経路のデータの作成および登録を実施します。

○OSQLファイルの作成

```
# su - seed
$ cd ~/work/db_setup
$ unzip N02-20_GML.zip
$ cd N02-20_GML
$ shp2pgsql -a -W cp932 -I -s 4326 N02-20_Station.shp station > Station.sql
→エラーとならずに処理が終了することを確認する。
$ ls -l Station.sql
→0バイト以上のファイルが存在することを確認する。
$ shp2pgsql -a -W cp932 -I -s 4326 N02-20_RailroadSection.shp railroad_section_work > Ra
ilroadSection.sql
→エラーとならずに処理が終了することを確認する。
```

```
$ ls -l RailroadSection.sql
```

→0バイト以上のファイルが存在することを確認する。

○OSQLファイルの登録

```
$ psql -U seed_usr -d seed_db -f Station.sql
```

```
$ psql -U seed_usr -d seed_db -f RailroadSection.sql
```

→エラーとならずに処理が終了することを確認する。

```
$ exit
```

5.5.4 駅・経路のデータ更新

以下のコマンドを実行し、鉄道における駅・経路のデータの更新を実施します。

○不要データの削除

```
# psql -U seed_usr -d seed_db
```

→seed_usrユーザのパスワードを入力し、ログインする

```
seed_db=# DELETE FROM station WHERE ST_YMin(ST_LineMerge(geom)) >= 37.88 OR S  
T_YMax(ST_LineMerge(geom)) <= 36.88;
```

→「DELETE 9837」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# DELETE FROM station WHERE n02_003 != '常磐線';
```

→「DELETE 402」が表示され、エラーとならずに処理が終了することを確認する。

○CSVファイルからデータの更新

```
seed_db=# CREATE TEMP TABLE tmp (n02_005 text, time integer, station_order integer);
```

→「CREATE TABLE」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# COPY tmp(n02_005,time,station_order) from '/home/seed/work/db_setup/train_mov  
e_time.csv' WITH CSV DELIMITER ',' header;
```

→「COPY 28」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# UPDATE station SET time = tmp.time, station_order = tmp.station_order FROM  
tmp WHERE station.n02_005 = tmp.n02_005;
```

→「UPDATE 28」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# DROP TABLE tmp;
```

→「DROP TABLE」が表示され、エラーとならずに処理が終了することを確認する。

○駅の中心を設定

```
seed_db=# UPDATE station SET point = ST_LineInterpolatePoint(ST_LineMerge(geom),0.5);
```

→「UPDATE 28」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# UPDATE station SET mag_geom = CAST(ST_Buffer(CAST(geom as geography),  
100, 'endcap=square join=round') as geometry);
```

→ 「UPDATE 28」が表示され、エラーとならずに処理が終了することを確認する。

○不要データの削除(railroad_section_work)

```
seed_db=# DELETE FROM railroad_section_work WHERE n02_003 != '常磐線';
```

→ 「DELETE 21753」が表示され、エラーとならずに処理が終了することを確認する。

○rail_nodeテーブルのデータ格納

```
seed_db=# INSERT INTO rail_node (geom) SELECT DISTINCT (a.g).geom FROM (SELECT ST_Dump(ST_Collect(ST_StartPoint(St_LineMerge(geom)), ST_EndPoint(St_LineMerge(geom)))) AS g FROM railroad_section_work) AS a;
```

→ 「INSERT 0 181」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX ON rail_node USING GIST (geom);
```

→ 「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

○railroad_sectionにsourceとtargetを設定

```
seed_db=# UPDATE railroad_section_work SET source = rail_node.id FROM rail_node WHERE ST_Equals(rail_node.geom, ST_StartPoint(St_LineMerge(railroad_section_work.geom)));
```

→ 「UPDATE 183」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# UPDATE railroad_section_work SET target = rail_node.id FROM rail_node WHERE ST_Equals(rail_node.geom, ST_EndPoint(St_LineMerge(railroad_section_work.geom)));
```

→ 「UPDATE 183」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX ON railroad_section_work (source);
```

→ 「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX ON railroad_section_work (target);
```

→ 「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

○stationにsourceとtargetを設定

```
seed_db=# UPDATE station SET source = rail_node.id FROM rail_node WHERE ST_Equals(rail_node.geom, ST_StartPoint(St_LineMerge(station.geom)));
```

→ 「UPDATE 28」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# UPDATE station SET target = rail_node.id FROM rail_node WHERE ST_Equals(rail_node.geom, ST_EndPoint(St_LineMerge(station.geom)));
```

→ 「UPDATE 28」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX ON station (source);
```

→ 「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# CREATE INDEX ON station (target);
```

→ 「CREATE INDEX」が表示され、エラーとならずに処理が終了することを確認する。

```
seed_db=# quit
```

○merge_railroad.pyを実行してデータ整形

```
# su - seed
$ cd ~/work/db_setup
$ unzip station_batch.zip
$ cd station_batch
$ source /home/seed/work/db_setup/module/SEED/env/bin/activate
$ python3 merge_railroad.py
→エラーとならずに処理が終了することを確認する。
$ deactivate
```

5.5.5 データのエクスポート

以下のコマンドを実行し、登録したデータをDMP形式でエクスポートします。

○データのエクスポート

```
$ cd ~/work/db_setup
$ pg_dump -U seed_usr -Fc seed_db > SEED_db.dmp
→seed_usrユーザのパスワードを入力し、エラーとならずに処理が終了することを確認する。
$ ls -l SEED_db.dmp
→0バイト以上のファイルが存在することを確認する。
```

— 以上 —