Interactive - 32 - only one a fixed set of rows. (pre-populate with key, pk, check-constraint on key)

There is a different way to implement a set of configuration items. In this case the items are stored one in each row in a set of fixed rows.

```
DROP  TABLE if exists ct_config_row ;

CREATE TABLE ct_config_row (
      id                serial not null primary key
    , name              text not null check ( name in (
                            'security_method',
                            'encryption'
                        ) )
    , ty                text not null default 'str'
    , value             text
    , i_value           bigint
    , b_value           boolean
);

CREATE UNIQUE INDEX ct_config_row_p1 on ct_config_row ( name );
```

With a table like this we can insert some values:

```
INSERT INTO ct_config_row ( name, value ) values
    ( 'security_method', 'jwt' ),
    ( 'encryption', 'es' )
;
```

The fixed set of configuration items is checked with the check constraint and the unique key on name.

A trigger can be used to prevent deletion of items.

```
CREATE OR REPLACE FUNCTION ct_config_row_prevent_delete()
RETURNS trigger AS $$
BEGIN
    IF OLD.config_id = 1 THEN
        RAISE EXCEPTION 'cannot delete configuration row';
    END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER ct_config_row_prevent_delete
    BEFORE DELETE ON ct_config_row
    FOR EACH ROW EXECUTE PROCEDURE ct_config_row_prevent_delete();
```

Tags: “fixed set rows”,“trigger”,“check constraint”

Validate: SQL-Select,“select ‘PASS’ as x”