Isok -- Query Based Data Integrity Management For PostgreSQL

Karl O. Pinc

Contents

List of Figures

List of Tables

1 A Very Short Introduction

Isok[1] is a PostgreSQL extension for monitoring and validating data using SQL queries.

Queries find problems. Isok builds on this and tracks *changes* in problematic data, changes that might or might not indicate a new problem. When configured to accept some questionable rows but not others of the same kind, Isok acts like a “soft trigger”, with scheduling and note-taking features to support the management of problem resolution.

Useful for "zero code required" batch-based data validation and data cleanup management. Most useful when review is required to determine whether a questionable data pattern should be allowed to remain in a database.

2 Why Use Isok?

Simplicity is one appealing property. Writing a query to find data problems is simple. Give the query to Isok and use it to manage your problem resolution process.

Do you ...

- Import data into PostgreSQL to be cleaned up later?
- Clean up database content over time?
- Allow or dis-allow specific data patterns on a case-by-case basis?
- Monitor data for changes, or for unusual but not dis-allowed conditions?
- Not want to write data validation apps or find triggers unsuitable, or too much work?

Isok may be for you if you are involved in data cleanup, or data integrity maintenance, or don’t want to put your data monitoring into an app or otherwise design something, or are tired of re-examining the "problems" your queries report that you have determined are not really problems.

Isok can help you manage your data’s integrity, especially when little technical effort can be spared or manual review is involved.

- If you can write a query to find problematic, but sometimes allowed, data, Isok will show you only those problem cases that you have not accepted as valid. You don’t have to repeatedly re-review query output.

[1] Isok <==> “Is” “ok”
Clever, right???
Har!

- If you want to manage your data cleanup over time, Isok can help ensure that newly added data is "cleaned", while scheduling the cleanup of old issues.
- If you can write a query to find problems in your data, don't want to engineer anything more, and want a system to track and manage the problems discovered.

Discover problematic data patterns, track them, and manage them. Manage issue resolution, which may involve accepting questionable data, unchanged. Report not only the *existence* of particular data patterns, but also report *changes* to values of previously accepted data. Isok is especially suited when importing "dirty" data into PostgreSQL for cleanup and analysis, and for corner cases where business logic is "fuzzy".

There can be a use-case to monitor for, and manage, outright errors in data, when you don't want to use triggers or, especially, constraints, for this purpose.[2] Isok can create, in effect, what may be thought of as a "soft trigger". One that is as simple to make as it is to write a query.

2.1 The One-to-One-Or-More Problem Almost Everyone Has

Most database schema designs will have at least one inter-table relationship that is One-to-One-Or-More. An example might be a CUSTOMERS table and an ORDERS, table where a customer may have multiple orders but is expected to have at least one.

The problem is that the ORDERS table typically has a CustomerID column, which contains a foreign key that references a CUSTOMERS row, and this is enforced with a foreign key constraint. The constraint requires a CUSTOMERS row to exist, before an ORDERS row can reference it. So the CUSTOMERS row is created first, and then rows are added to ORDERS.

But this opens up the possibility for a a customer to exist without having any related orders. Now, this may or may not be desirable.[3] Or, there may even be time limits on how long such a condition should be allowed to exist. After a certain amount of time, you may want to remove from the database a customer that has never ordered anything. Regardless, Isok provides a simple way to manage the situation.

Isok can be given a query like:

```
SELECT customers.id
     , 'Customer ' || customers.id || ' has no related ORDERS'
     , NULL
  FROM customers
  WHERE NOT EXISTS (SELECT 1
                      FROM orders
                      WHERE orders.customerid = customers.id);
```

[2] Triggers and constraints are the usual data validation methods, because these prevent erroneous data from getting into the database in the first place. But you may need all data, "valid" or not, to be in your database, or you may have other reasons why triggers or constraints are not an appropriate approach.

[3] Should you *never* want customers without orders, you can mark your referential constraint as `INITIALLY DEFERRED`, or, in older PostgreSQLs, create an initially deferred constraint trigger.
The application that creates customers and orders must then put the creation of a customer, with an initial order, into a transaction.

With this query, Isok's features help manage customers that have no related orders.

3 How Isok Works

Isok is a PostgreSQL extension for monitoring anything that can be reported with an SQL query. Unlike simply running a query, which reports the *existence* of questionable data patterns, Isok produces reports alerting you of *changes* to questionable data patterns, so that only new problems need be reviewed.

In this way, Isok makes practical the monitoring and management of unusual, but sometimes allowed, data patterns. It has record-keeping and scheduling features to assist with the management of problem resolution over time.

To use Isok you write SQL queries that produce reports alerting you of questionable data patterns. Then, on a row by row basis, you can defer, possibly forever, the appearance of individual alerts on future reports.

Isok is useful to periodically probe for unusual but allowed activity, such as the addition of a new country code. Or the purchase of more than 1,000 shoes by one person. Approved excessive shoe purchases can be individually flagged so they do not appear in future reports. To avoid being overwhelmed by numerous legitimate alerts and to allow time to resolve issues, specific rows in the reports can be deferred so they do not reappear before a designated date.

Unlike triggers and constraints, Isok does nothing until executed.[4] This is done by **SELECT**ing **FROM** a function, which runs some or all of the saved queries to check the state of the database and report the results. Report content is archived and can be queried.

A reported issue, a row returned by a saved query, is classified as either an error or a warning. Errors are always reported when Isok is run. After execution, the warnings reported by the user-supplied queries may be manually sorted by the Isok user into one of the following categories: unclassified (the default), labeled "resolved", or deferred until a later date. When Isok is run, unclassified warnings are reported, "resolved" warnings are not reported, and deferred warnings are not reported until the current date reaches the deferral date.

PostgreSQL supports a high degree of introspection. Isok can therefore monitor PostgreSQL itself, both the database engine's operational metrics and database schema design. In the former case system performance or usage might be monitored. In the latter, monitoring might look for things like violations of column naming conventions. However, while there may be legitimate uses of Isok for these sorts of purposes, other tools may be a better fit.

Regardless of how Isok is used, we believe some monitoring or some error checking is better than no monitoring and no error checking. Isok makes monitoring and error checking easy. If introducing triggers into your processing or running a complete performance monitoring solution is just not feasible, Isok provides a simple way to move the ball at least a little bit closer to the goal.

[4] There is an appendix on automating Isok execution.

4 A Start-To-Finish Set of Examples

These examples demonstrate looking for unexpected new country codes, or for the unexpected disappearance of an existing country code.

The examples, or at least the installation related portions, must be run on the machine that is running the PostgreSQL server, the server running the database's backend engine.

Each example expects the code shown in previous examples to have been executed.

4.1 The OS Side of Isok Installation

Begin by installing Isok into the OS, and connecting to a database.

```
$ #
$ # Install Isok
$ #
$
$ # Install the required shell commands
$ # (If you are on an RPM based system, use the `dnf` command instead of
$ # the `apt` command.)
$ sudo apt install make ❶
<uninteresting output redacted>
$ sudo apt install curl
<uninteresting output redacted>
$ sudo apt install unzip
<uninteresting output redacted>
$
$ # Pick your Isok version
$ export ISOK_VERSION=0.4.1
$
$ # Download pg_isok and install into the OS so the db engine can find it.
$ curl https://api.pgxn.org/dist/pg_isok/${ISOK_VERSION}/pg_isok-${ ↩
    ISOK_VERSION}.zip > pg_isok-${ISOK_VERSION}.zip
<uninteresting output redacted>
$ unzip pg_isok-${ISOK_VERSION}.zip
<uninteresting output redacted>
$ cd pg_isok-${ISOK_VERSION}
$ sudo make install
<uninteresting output redacted>
$
$ #
$ # Interact with a database
$ #
$ psql ❷
psql (15.13 (Debian 15.13-0+deb12u1))
```

```
Type "help" for help.

=> \pset pager ❸
Pager usage is off.
=>
```

❶ The **sudo** command is used here by way of example. The point is, these commands must be run with elevated permissions.

❷ The **psql** command may well need additional arguments supplied in order to connect to the right database server, to the right database, as the right user, and so forth.

❸ For purposes of the example, send all output directly to the screen, rather than to an interactive pager.

4.2 Database Setup

Next, install Isok and create some sample data to be used for testing. At the end of this step we will be ready to work with Isok.

```
=> --
=> -- Install pg_isok, in a schema called "isok"
=> --
=> CREATE SCHEMA isok;
CREATE SCHEMA
=> CREATE EXTENSION pg_isok SCHEMA isok;
CREATE EXTENSION
=>
=> --
=> -- Set up a "workspace" for the example, with sample data
=> --
=> SET search_path TO workspace, isok; ❶
SET
=> CREATE SCHEMA workspace;
CREATE SCHEMA
=> CREATE TABLE countries
     (code TEXT PRIMARY KEY, description TEXT NOT NULL);
CREATE TABLE
=> INSERT INTO countries (code, description)
     VALUES ('oc', 'Oceania')
          , ('ea', 'Eastasia')
          , ('eu', 'Eurasia');
INSERT 0 3
=>
```

❶ This is for convenience, so that table names, and so forth, do not have to be qualified with the schema in which the table exists.

Because the search path begins with `workspace`, by default, new tables are created there.

4.3 Our First Query

Our first goal is to configure Isok so that it will tell us when a new country is put into the system. To do that, we give Isok a query that searches for new countries. Then, we see how to use Isok to run the query, and see what running it does.

The queries given to Isok must return three columns.

QR_ID A value that is, per-query, unique.

This value, together with the the query identifier, is used as the key to identify a specific reported problem. You will need to use the key to work with particular problems. For example when telling Isok to "resolve" some problem, to prevent the problem from appearing on future reports.

QR_Message Text that fully describes the problem.

QR_Extra Any other information about the problem that needs to be stored. Only more advanced users will want to return a value in this column. Most of the time your queries will return `NULL` in this column.

As in the example below, when writing **INSERT** statements to save your queries in Isok it is best to use dollar quoting to quote your queries.

```
=> --
=> -- Prepare Isok for use
=> --
=>
=> -- Create the vocabulary used to classify Isok queries.
=> INSERT INTO iq_types (iqtype, description)
     VALUES ('code_chk', 'Check the system''s codes');
INSERT 0 1
=> -- Save a query that looks for new country codes.
=> INSERT INTO isok_queries (iqname, error, type, keep, query, comment)
     VALUES('new_countries'    -- iqname
          , false              -- error ❶
          , 'code_chk'         -- type, from the IQ_TYPES table
          , true               -- keep ❷
          , $$SELECT countries.code ❸
                   , 'Unexpected new country in COUNTRIES: Key (Code) = ('
                     || countries.code
                     || '), Value (Description) = ('
                     || countries.description
```

```
                    || ')' ❹
                  , NULL ❺
               FROM countries
               ORDER BY countries.code$$ ❻                  -- query
         , 'Find all the countries, identify them by code' -- comment
          );
INSERT 0 1
=>
=> --
=> -- Initial run of Isok, show all the "new" countries.
=> --
=> SELECT riq.irid, riq.iqname, riq.type, riq.keep_until
        , riq.qr_id, riq.qr_message, riq.qr_extra
     FROM run_isok_queries($$VALUES ('new_countries')$$) ❼
       AS riq
     ORDER BY riq.iqname, riq.qr_id;
 irid |    iqname     | type | keep_until | qr_id |  ↩
                                           qr_message  ↩
                                           | qr_extra
------+---------------+------+------------+-------+-----------------------------------------

    2 | new_countries |      | infinity   | ea    | Unexpected new country ↩
         in COUNTRIES: Key (Code) = (ea), Value (Description) = (Eastasia) ↩
         |
    3 | new_countries |      | infinity   | eu    | Unexpected new country ↩
         in COUNTRIES: Key (Code) = (eu), Value (Description) = (Eurasia)  ↩
         |
    1 | new_countries |      | infinity   | oc    | Unexpected new country ↩
         in COUNTRIES: Key (Code) = (oc), Value (Description) = (Oceania)  ↩
         |
(3 rows)

=> -- See that the above results have been saved in ISOK_RESULTS.
=> SELECT isok_results.irid, isok_results.iqname, isok_results.type
        , isok_results.keep_until, isok_results.qr_id
        , isok_results.qr_message, isok_results.qr_extra
     FROM isok_results
     ORDER BY isok_results.iqname, isok_results.qr_id;
 irid |    iqname     | type | keep_until | qr_id |  ↩
                                           qr_message  ↩
                                           | qr_extra
------+---------------+------+------------+-------+-----------------------------------------

    2 | new_countries |      | infinity   | ea    | Unexpected new country ↩
         in COUNTRIES: Key (Code) = (ea), Value (Description) = (Eastasia) ↩
         |
    3 | new_countries |      | infinity   | eu    | Unexpected new country ↩
```

```
         in COUNTRIES: Key (Code) = (eu), Value (Description) = (Eurasia)  ↩
         |
    1 | new_countries |      | infinity   | oc    | Unexpected new country ↩
         in COUNTRIES: Key (Code) = (oc), Value (Description) = (Oceania)  ↩
         |
(3 rows)

=>
```

1. The result rows the query produces are not errors, they are warnings. Errors are not very interesting, reports always show errors. But interesting things can be done with warnings, as will be seen.

1. Keep the result rows in ISOK_RESULTS, even if the row is not (re)produced when the query is re-run. The utility of this should become clear below.

3. The country code is unique, among the query results produced by the `new_countries` query being created here, and so can be used when querying ISOK_RESULTS to uniquely identify any given row of this query's output.

   This value is saved in ISOK_RESULTS.QR_ID. We will use the fact that it is a `COUNTRIES.Code` value later in the example.

   Tip
   While this example query generates a simple ISOK_RESULTS.QR_ID, often something more complex is needed to ensure these ids have the two necessary requirements, per-query uniqueness and reproducibility.
   Concatenating multiple values, separated by some delimiter (like "*"), can be a good approach. Alternately, instead of using a delimiter, embedding the same multiple values within a descriptive string is sometimes useful. This approach can bring two levels of detail to your reporting, with the core problem in ISOK_RESULTS.QR_ID and further information in ISOK_RESULTS.QR_Message.

4. The explanatory message that makes clear why the row is showing up as a warning.

   This value is saved in ISOK_RESULTS.QR_Message.

5. This value is saved in ISOK_RESULTS.QR_Extra.

6. It is good practice to write your queries to **ORDER BY** something unique, so that the results are always delivered in a consistent order.

7. Because there's only one query, we could simply not give run_isok_queries() an argument, invoking it as **run_isok_queries()**. This runs "all queries", that is, the single query we have. But it seems better practice to be explicit and pass run_isok_queries() the query we want to run.

4.4 Resolving Warnings

We've seen, above, that the first time Isok runs our query, it reports that every country is a new country. But the countries we have are the countries we want, so we want to resolve the issues reported with our existing countries.

To resolve each reported warning, we tell Isok that we never want it to report the warning again. ISOK_RESULTS is where our reported problems are stored, as rows in the table. We mark each row produced by the `new_countries` query, telling Isok that we never want to see the row again. This is done by setting ISOK_RESULTS.Deferred_To to `infinity`. With that done, Isok won't show us the rows until the end of time.

After resolving our issues the system is "mature", in that Isok has been integrated into our operations and its tables reflect the current state of the database. It knows that the countries that already exist should exist, and no longer reports them as problems.

```
=> --
=> -- Tell Isok that the "new" countries are all acceptable, we don't
=> -- need to see them when looking for problems.
=> --
=> UPDATE isok_results
     SET deferred_to = 'infinity' ❶
     WHERE iqname = 'new_countries';
UPDATE 3
=>
=> -- The accepted countries don't show when we re-run the query.
=> SELECT riq.irid, riq.iqname, riq.type, riq.keep_until
        , riq.qr_id, riq.qr_message, riq.qr_extra
     FROM run_isok_queries($$VALUES ('new_countries')$$)
       AS riq
     ORDER BY riq.iqname, riq.qr_id;
 irid | iqname | type | keep_until | qr_id | qr_message | qr_extra
------+--------+------+------------+-------+------------+----------
(0 rows)

=> --
=> -- New countries show up when we re-run our query, but not the ones
=> -- we've accepted.
=> --
=>
=> -- Insert a new country
=> INSERT INTO countries (code, description)
     VALUES ('mv', 'Margaritaville');
INSERT 0 1
=>
=> -- Run our query again, looking for problems. ❷
=> SELECT riq.irid, riq.iqname, riq.type, riq.keep_until
        , riq.qr_id, riq.qr_message, riq.qr_extra
```

```
     FROM run_isok_queries($$VALUES ('new_countries')$$)
       AS riq
     ORDER BY riq.iqname, riq.qr_id;
 irid |    iqname     | type | keep_until | qr_id |  ↩
                                             qr_message  ↩
                                             | qr_extra
------+---------------+------+------------+-------+-----------------------------------------------

   10 | new_countries |      | infinity   | mv    | Unexpected new country ↩
        in COUNTRIES: Key (Code) = (mv), Value (Description) = ( ↩
       Margaritaville) |
(1 row)

=>
```

❶ Using `infinity` tells Isok that we never want to see the warning again. The warning is *resolved*.

If, instead, you wanted to put off figuring out what to do about some particular warning, you could defer its reporting until some future date.

❷ The exciting part here, the whole point of the exercise, is that the countries we started out with *don't* re-appear in the report. Their existence only has to be reviewed once, no matter how many time the saved query is re-run.

❸ The new country shows up on our problem report.

4.5 A Query That Looks For Missing Countries

How do we detect that a country has gone missing?

After Isok has run the `new_countries` query at least once, the ISOK_RESULTS.QR_ID column contains every existing country code. And, because we set the the `new_countries` query's ISOK_QUERIES.Keep value to `infinity`, the `new_countries` query's rows are not removed from ISOK_RESULTS, even when the query no longer returns the row. So, we can query ISOK_RESULTS, comparing it to what's in the `COUNTRIES` table, to find country codes that should exist, but don't. This section of the example does exactly that.[5]

[5] Because Isok can run multiple queries in a single invocation, the careful reader might wonder whether interactions between queries can produce inaccurate results. Indeed, if a query in ISOK_QUERIES references ISOK_RESULTS, there can be interactions.

In the case we're considering, detecting deleted countries, it does not matter.

In other cases, it is *possible* that some query might put rows into ISOK_RESULTS, confusing a query run afterward that uses ISOK_RESULTS, although it is hard to imagine such a situation. In any case, should query ordering matter, testing ISOK_RESULTS.Last_Seen against `CURRENT_TIMESTAMP`, which remains constant throughout an execution of run_isok_queries(), might help avoid the problem.

```
=> --
=> -- Show how to detect deleted countries
=> --
=>
=> -- Give Isok a query that finds deleted countries
=>
=> INSERT INTO isok_queries (iqname, error, type, keep, query, comment)
     VALUES('deleted_countries'   -- iqname
          , false                 -- error
          , 'code_chk'            -- type, from the IQ_TYPES table
          , false                 -- keep
          , $$SELECT isok_results.qr_id ❶
                  , 'Unexpected deletion from COUNTRIES: Key (Code) = ('
                    || isok_results.qr_id
                    || ')'
                  , NULL
                FROM isok_results
                WHERE isok_results.iqname = 'new_countries'
                  AND NOT EXISTS ❷
                    (SELECT 1
                       FROM countries
                       WHERE countries.code = isok_results.qr_id)
                ORDER BY isok_results.qr_id$$               -- query
          , 'Report deleted COUNTRIES.Code values'          -- comment
           );
INSERT 0 1
=>
=> -- Delete a country
=> DELETE
     FROM countries
     WHERE code = 'eu';
DELETE 1
=>
=> -- Run both queries, to find both the new country and the deleted  ↩
    country.
=> SELECT riq.irid, riq.iqname, riq.type, riq.keep_until
        , riq.qr_id, riq.qr_message, riq.qr_extra
     FROM run_isok_queries($$VALUES ('new_countries')
                                  , ('deleted_countries')
                               ORDER BY 1$$) ❸
       AS riq
     ORDER BY riq.iqname, riq.qr_id;
 irid |      iqname       | type | keep_until | qr_id |  ↩
                                                qr_message  ↩
                                                | qr_extra
------+-------------------+------+------------+-------+-----------------------------------
```

```
   18 | deleted_countries |      |             | eu     | Unexpected  ↩
      deletion from COUNTRIES: Key (Code) = (eu) ❹  ↩
                                               |
    9 | new_countries     |      | infinity    | mv     | Unexpected new  ↩
      country in COUNTRIES: Key (Code) = (mv), Value (Description) = ( ↩
      Margaritaville) |
(2 rows)

=>
```

❶ Again, the country code is unique, among the query results produced by the `deleted_countries` query, and so is a suitable key component.

❷ Here, we rely on having set the ISOK_QUERIES.Keep flag in the `new_countries` query. It caused the ISOK_RESULTS.Keep_Until column to be set to `infinity`, so that the ISOK_RESULTS rows are not deleted even when the query no longer returns them.

Because the ISOK_RESULTS rows remain, we are able to use their existence to test for deletion of `COUNTRIES` rows. If the ISOK_RESULTS.QR_ID were not a plain `COUNTRIES.Code` value, doing this might require more ingenuity. But this sort of thing should always be possible, given a little forethought.

❸ Without the `ORDER BY` the order in which the queries are executed is undefined, and therefore the ordering of the results returned is not guaranteed to be consistent.

❹ With some ingenuity, likely involving the ISOK_RESULTS.QR_Extra column, the message could be made more informative. Whether this is worth doing is up to the reader.

5 Installation

There are two steps to installation. First, getting and preparing the code and, when installing as an extension, installing into the OS, and, second, loading into one or more databases.

Note

Installing Isok into the OS, which is necessary when installing as an extension, typically requires elevated OS-level privileges, such as `root` privileges. The examples given do not include the assumption of elevated privileges, or show the use of any particulars, such as the **sudo** command, needed to assume such privileges.

Similarly, the examples do not include the connection parameters (usernames, passwords, hosts, database names, etc.) which may be needed to connect to a database.

Regardless of how Isok is installed, we recommend you install it in a dedicated `schema`. Dedicating a schema to Isok has a number of benefits, not the least of which is simplified access control to mitigate security concerns.[6] When a schema is created, only the owner can access its content. This is sufficient protection, assuming care is taken using the ISOK_QUERIES.Role and ISOK_QUERIES.Search_Path columns. (Or, if these features are disabled.)

5.1 Requirements

Isok installs on PostgreSQL version 10 or later, although PostgreSQL versions no longer supported by The PostgreSQL Global Development Group may get reduced support.

5.2 Quick-Start

Most people should do a normal install, which installs Isok as an extension.

If you're running a managed instance of PostgreSQL, in the cloud or otherwise, and don't have access to the machine running the PostgreSQL cluster, you'll need to install from SQL.

5.2.1 Normal Install

- Login to the machine running your PostgreSQL database
- Download the pg_isok distribution
- **unzip pg_isok-*.zip**
- **cd** into the directory produced
- **sudo make install**
- Execute SQL like:

```
CREATE SCHEMA isok;
CREATE EXTENSION pg_isok SCHEMA isok;
```

[6] The downside to installing in a schema is that when writing SQL you must either set your **search_path** or qualify names, by prepending the object name with the schema name and a period. For example, without setting a search path, if you installed into a schema named **isok**, instead of **SELECT * FROM run_isok_queries();**, you would have to write **SELECT * FROM isok.run_isok_queries();**.

5.2.2 SQL Install

Download the Isok zip file, unzip it, and **cd** into its directory.

Choose the name of the schema into which you wish to install Isok. Generate Isok's SQL with:

```
make TARGET_SCHEMA=myschema sql/pg_isok_cloud--VERSION.sql
```

Where `VERSION` is the version of Isok you are installing.

Create the schema you've chosen, if it does not exist, and execute the SQL found in the `sql/pg_isok_cloud--VERSION.s` file.

5.3 Preparing, While Logged-In to Un*x

Installing as an extension requires that the installation be done while logged into the PostgreSQL server's machine. Or, at minimum, while the current working directory is within the server's filesystem.

Installing from SQL, as is necessary when the PostgreSQL's server's filesystem is unavailable, must be done from a machine able to work as a PostgreSQL client.

When installing from SQL, the recommended download is the Isok zip file "distribution" from PGXN.org. It is "pre-built", and so does not require installation of any build tooling. If you have this, after unzipping, you can skip over the next sections, which cover disabling features, and cloud installation, and skip straight to Section 5.3.3.

It is also possible to clone the Isok git repository, but be forewarned. Working from the git repository requires the installation of considerable tooling.

Note

Any rebuild of Isok requires the installation of the **m4** macro pre-processor.[a] Your operating system almost surely makes available an `m4` package.
Only the "pre-built" PGXN distribution can be installed without the use of **m4**.

---

[a] Possibly, the GNU m4 implementation is required. This is what PostgreSQL requires, and alternatives have not been tested.

Note

One would expect to be able to use pgxnclient for installation. However, it has limitations that make it difficult to use.

5.3.1 Re-Building to Disable Features

If desired, some potentially dangerous features of Isok can be disabled at build time.

These are the **make** variables that control the build options:

DISABLE_ROLE Disable the ability to **SET ROLE** from ISOK_QUERIES.

DISABLE_SEARCH_PATH Disable the ability to **SET** the **search_path**.

To use these variables, set them to any value when running **make**. For example, to disable all optional features run:

```
make DISABLE_ROLE=y DISABLE_SEARCH_PATH=y
```

The build configuration is documented in the **doc/pg_isok--${VERSION}.config** file, and installed with the rest of the documentation.

5.3.2 Building for and Installing in The Cloud (Installing From SQL)

If you are running in the cloud, or some other managed instance where you do not have permissions on the host running PostgreSQL, you will not be able to install Isok as an extension. In these cases you can still install Isok, but you must first build its SQL and then manually execute it.

Of course, this installation method can always be used, as there is always a way to execute SQL.

To build a "cloud version" of Isok, suitable for installation by SQL execution, you would type something like:

```
make TARGET_SCHEMA=isok pg_isok_cloud--$(cat VERSION).sql
```

The resulting sql file is in the **sql/** directory.

To customize the build, any of the above variables may also be set. The **TARGET_SCHEMA** variable must be set; the objects produced by the generated SQL must be located within a designated `schema`. It is highly recommended that the **TARGET_SCHEMA** be lower-case and otherwise be a PostgreSQL name which does not require quoting.

To install, first create the `schema` and then execute the sql. The command line interaction, if you use the **psql** command line client interface, would look something like:

```
$ psql
psql (15.13 (Debian 15.13-0+deb12u1))
Type "help" for help.

me=> CREATE SCHEMA isok;   -- The TARGET_SCHEMA used to build the sql
CREATE SCHEMA
```

```
me=> \i sql/pg_isok_cloud--1.0.0.sql
<lots of output redacted>
me=> \q
$
```

You must re-build different SQL, with a different **TARGET_SCHEMA**, to install a second instance of Isok into a different schema.

5.3.3 Installing in the PostgreSQL Server's OS

With appropriate OS-level permissions, run:

```
make install
```

With this step complete, you are ready to install the Isok extension into any schema of any database in the cluster.

If you have more than one cluster installed on the machine, Isok is installed into the default cluster, the one reported on if you execute **pg_config**.

To install into a different cluster you must first find the **pg_config** command belonging to the cluster you wish to install into. The output of **pg_config --bindir** may be helpful in this regard. It shows the path to the default cluster's **pg_config**.

Having found the full path to the **pg_config** command of the target cluster, install Isok into that cluster (as `root`) with something like:

```
make PG_CONFIG=/usr/lib/bin/postgresql/15/bin/pg_config ↩
    install
```

5.3.4 Running Regression Tests

Once an extension has been installed in the OS, regression tests can be run to test whether Isok is operating correctly. Running the regression tests when Isok is installed by SQL execution is unsupported.

The same build variables must be set when running the regression tests as when the system was built. (The PGXN distribution sets no variables, the default.) Should you set a different collection of variables than when building, some tests will fail and others may fail to run at all.

The following example runs the default set of regression tests:

```
make installcheck
```

5.4 Loading Into PostgreSQL

The **CREATE EXTENSION** command is used to install Isok, as in the following example:

```
CREATE SCHEMA isok;
CREATE EXTENSION pg_isok SCHEMA isok;
```

5.5 Uninstalling

5.5.1 Uninstalling From the OS

Uninstalling from the OS does the opposite of installing. It removes the extension from the PostgreSQL server's filesystem.

To uninstall using **make**, run:

```
make uninstall
```

To uninstall with `pgxnclient`, run:

```
pgxn uninstall pg_isok
```

Because Isok is pure SQL, uninstalling it from the OS does not remove any functionality from existing instances installed with **CREATE EXTENSION**. Uninstalling does, however, remove the ability to use the **CREATE EXTENSION** to install Isok in a database.

5.5.2 Uninstalling From PostgreSQL

Running:

```
DROP EXTENSION pg_isok;
```

removes the extension from all schemas in the current database.

To remove an installation of Isok from an individual schema, drop the schema with **DROP SCHEMA schemaname CASCADE;**.

6 An Overview of the Tables

This section provides an overview of Isok's tables.

Table	One row for each...
ISOK_QUERIES	query used to discover data integrity problems
ISOK_RESULTS	data integrity problem discovered by Isok

Table 1: The Isok Tables

Table	Id Column	Related Column(s)	One entry for every possible choice of...
IQ_TYPES	IQType	ISOK_QUERIES.Type	kind of problem with data integrity
IR_TYPES	IRType	ISOK_RESULTS.Type	remark which might apply to more than one instance of questionable database integrity

Table 2: The Isok Support Tables

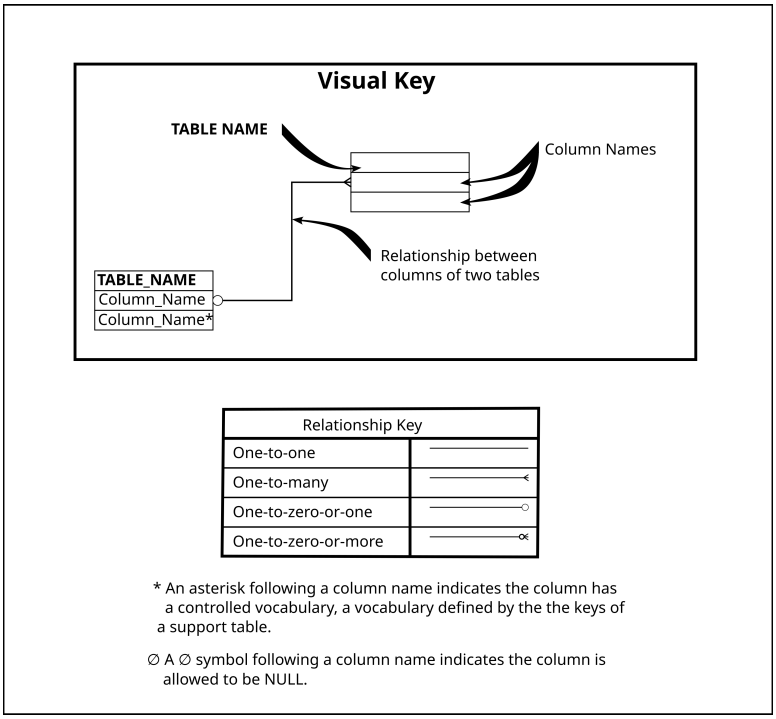

Figure 1: Key To Entity Relationship Diagrams

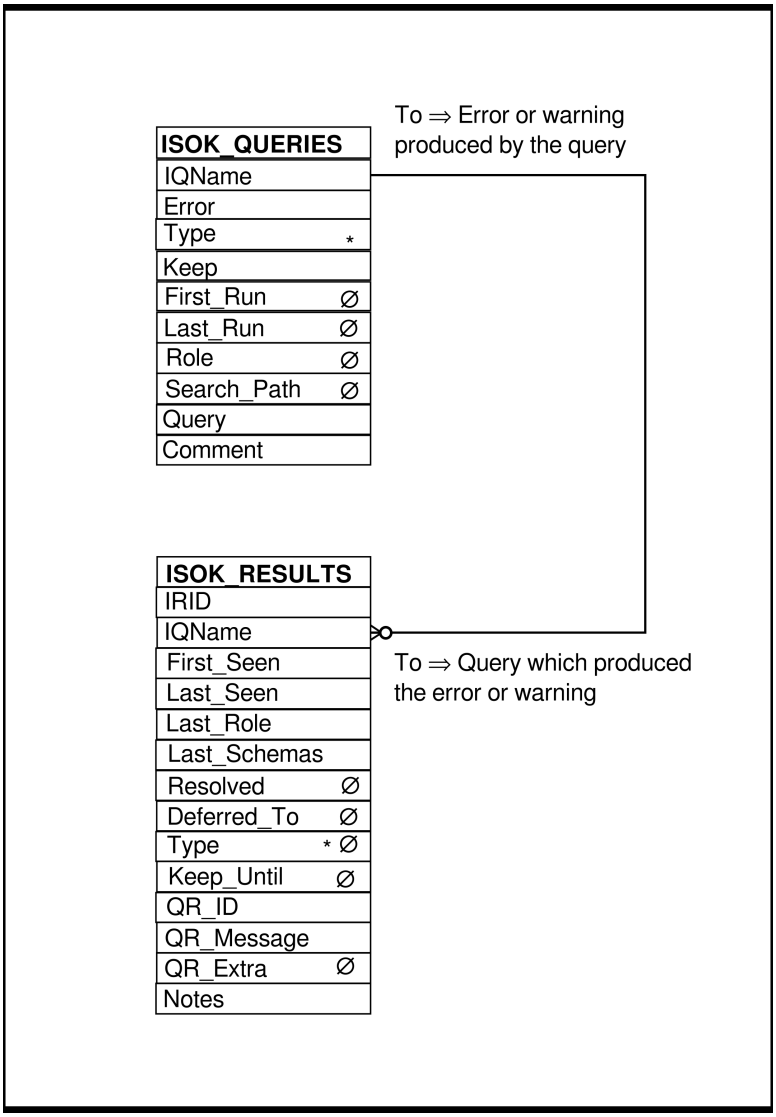

Figure 2: Isok Entity Relationship Diagram

7 The Main Tables

In the table descriptions below, each table has it's own section, with sub-sections for the table's columns.

All timestamps (date plus time values) have a one second precision. Fractions of a second are not recorded.

All timestamps track the time zone.

7.1 ISOK_QUERIES

The ISOK_QUERIES table contains one row for every query used to search for database integrity issues.

The Last_Run value cannot be before the First_Run value.

Tip

Use PostgreSQL's dollar quoting when inserting queries into ISOK_QUERIES using **INSERT** statements. This avoids problems that would otherwise arise involving the use of quote characters inside quoted strings.

Example 7.1 Inserting a query into ISOK_QUERIES using dollar quoting

```
-- Report a warning when there's a birth date before 1950
INSERT INTO isok_queries (iqname, error, type, keep, query,  ↩
    comment)
  VALUES('mycheck', false, 'bdate', false
       , $$SELECT 'Bad birth date: ' || mytable.id || ', ' ||  ↩
          mytable.birthdate
                  AS id
                , 'Id ('
                  || mytable.id
                  || ') has a birthdate ('
                  || mytable.birthdate
                  || ') before 1950'
                  AS msg
             FROM mytable
             WHERE mytable.birthdate < '1950-01-01'$$
       , $$Report a warning when there's a birthdate before 1950 ↩
          $$
        );
```

7.1.1 IQName (Isok Query Name)

A TEXT value. A unique name for the query. The IQName value cannot be changed. This column may not be empty; it must contain characters, and it must contain at least one non-whitespace character. This column may not be NULL. This column may not contain whitespace characters. This column must be unique when compared in a case-insensitive fashion.

7.1.2 Error

A BOOLEAN value. TRUE when the query finds conditions that are errors, FALSE when the query finds conditions that are warnings. See ISOK_RESULTS (and Section 3) for more on warnings and errors.

This column may not be NULL.

7.1.3 Type

A TEXT value. Code classifying the query. The legal values for this column are defined by the IQ_TYPES support table.

This column may not be NULL.

7.1.4 First_Run

A timestamp. Date and time the query was first run by Isok. The value of this column is NULL if the query has never been run.

7.1.5 Last_Run

A timestamp. Date and time the query was most recently run by Isok. The value of this column is NULL if the query has never been run.

7.1.6 Keep (Keep old results)

A BOOLEAN value. This column controls the value placed in the ISOK_RESULTS.Keep_Until column when run_isok_queries() inserts new rows in ISOK_RESULTS.

When this column is TRUE, each row returned by the query is stored in ISOK_RESULTS with a Keep_Until value of infinity. This prevents run_isok_queries() from deleting the query result row when run, when the query no longer returns the result row.

When this column is FALSE, the ISOK_RESULTS.Keep_Until value of any new rows that run_isok_queries() inserts is NULL.

This column may not be NULL.

7.1.7 Role

A PostgreSQL `name` value. The PostgreSQL role to use to run the query.

Because different roles have differing access to database content, it can be useful to run queries with different roles in effect.

Caution
Setting the role may have security implications.

This column is not validated against existing roles.

Note that the `name` data type casts (transparently) to `TEXT`.

When this column is `NULL`, the effective role is not changed.

7.1.8 Search_Path

A `TEXT` value. The PostgreSQL `schema` **search_path** to have in effect when the query is run.

The syntax of the search path is that used by **SET search_path ...** and returned by **SHOW search_path;**.

Because queries may not always contain schema names to qualify database objects, a single query can return different results depending on the **search_path** in effect. So it can be useful to run different queries when different schema search paths are in effect.

Caution
Setting the search_path may have security implications.

Care must be taken when setting the search path because the search path can be set to anything, regardless of which schemas exist or are available to the user. It is quite easy to set a search path that searches no schemas. PostgreSQL will not produce any warnings or errors should you do so.

When this column is `NULL`, the schema search path is not changed.

7.1.9 Query

A `TEXT` value. A query which checks for database integrity violations. The query need not end in a semi-colon. The query must return 3 columns. Although these columns are referred to by name below, the names the query gives to the columns does not matter.

Although the query executed by Isok must return three columns, there are different approaches to take regarding how the content of the first two columns are best structured.

7.1.9.1 The first returned column, the ID column

The first column is used as an id. It must contain a unique value. (Unique per results returned by the given query). The value must also be constant; repeated runs of the query which find the same problem must return a consistent value.

Caution

The system cannot enforce the requirement that the first column be consistent over repeated runs of the query. If the query does not satisfy this requirement Isok will generate duplicates of previously reported problems.

The value of the first column may not be `NULL` or the empty string.

Guidelines for the value of the first column are that it should be human readable and relatively short. It should probably contain id values in order to ensure uniqueness, but only those that will not change over time.

The value of this first column may need to be typed in or otherwise referenced by a person in order to make notes regarding the problem or to change the problem's status.

7.1.9.2 The second returned column, the Msg column

The second column contains a message describing the discovered database integrity problem. It should contain a complete description of the problem and may be as verbose as necessary.

The value of the second column may not be `NULL` or the empty string.

7.1.9.3 The third returned column, the Extra JSON column

The third column contains JSON data. The purpose of this column is to hold additional data on the reported condition that may need to be tracked, or queried. PostgreSQL is able to efficiently query `JSONB` data, which is how this column is stored.

Warning
At the time of this writing, in practice, returning a third column is optional. But this behavior should not be relied upon.
Best practice is to return a `NULL` value for the third column when you do not wish to store any JSON with the query result.
When only 2 columns are returned, the effect is the same as returning a `NULL` value in the third column. The third column is optional, in practice, because a portion of the PostgreSQL PL/pgSQL language is unspecified.[a] Isok cannot feasibly use the text of the Query column to determine how many columns the query returns. So it cannot prevent the query from being written to return only two columns. And, when this is the case, the present PL/pgSQL implementation allows the Query to return two columns instead of three.

[a] The unspecified PL/pgSQL behavior being, that the *target* in a PL/pgSQL statement of the form "**FOR** *target* **IN EXECUTE** *text_expression* **[USING** *expression* **[, ...]] LOOP**" is allowed to contain more variables than the *test_expression* returns columns, in which case the extra variables are assigned the `NULL` value. Because this behavior is undocumented, it is subject to change. Should this behavior change, returning a third column will be required, not optional.

Return a `NULL` value in the third column when there is no JSON data.

7.1.10 Comment

A `TEXT` value. A comment on the query. This may be as verbose as necessary. This column may not be `NULL`. This column may be empty; it need not contain characters, but it may not contain only whitespace characters.

7.2 ISOK_RESULTS (Isok query Results)

The ISOK_RESULTS table contains one row for every database integrity problem discovered by the queries in ISOK_QUERIES. That is to say, one row for every row returned by executed queries. The table's purpose is twofold. It provides an efficient way to list data integrity problems, without having to execute the potentially complex queries which discover the problems. But it's main purpose is to allow warnings, i.e. those problems discovered by the queries saved in ISOK_QUERIES rows having a `FALSE` Error value, to be resolved -- permanently marked as acceptable conditions. Resolved warnings can be safely ignored thereafter, and since Isok automatically ignores resolved warnings those responsible for maintaining database integrity need not repeatedly concern themselves with resolved conditions.

To resolve a warning place a timestamp in the Resolved column.

Data integrity errors can not be resolved, the erroneous data condition must be fixed -- ISOK_RESULTS rows must have a `NULL` Resolved value when the row has a IQName related to an ISOK_QUERIES row having a `TRUE` Error value.

The Last_Seen value, the Resolved value, and the Deferred_To value cannot be before the First_Seen value.

A resolved warning cannot be deferred -- either Resolved or Deferred_To, or both, must be NULL.[7]

The query result id generated by the stored query must be unique per query -- the combination of ISOK_RESULTS.IQName and ISOK_RESULTS.QR_ID must be unique.

7.2.1 IRID (Integrity Results Identifier)

An BIGINT value This column uniquely identifies the row containing the result of a database integrity query. The IRID value cannot be changed and is automatically generated with a PostgreSQL sequence.

7.2.2 IQName (Integrity Query Name)

A TEXT value. The ISOK_QUERIES.IQName value identifying the query which produced the result.

7.2.3 First_Seen

A timestamp value. Date and time the query result was first produced by Isok. This column may not be NULL.

7.2.4 Last_Seen

A timestamp value. Date and time the query result was most recently produced by Isok. This column may not be NULL.

7.2.5 Last_Role

A PostgreSQL name value. The role (user) which was the current role when the query was last executed. Note that the name data type casts (transparently) to TEXT.

This column is not validated against existing roles.

This column may not be NULL.

7.2.6 Last_Schemas

An array of PostgreSQL name values. All schemas that were, implicitly or not, in the **search_path**, and also available to the Last_Role, when the result was returned. For more information, see the documentation of the current_schemas() function.

Note that the name data type casts (transparently) to TEXT. This column may not be NULL.

[7] To remove an ISOK_RESULTS.Deferred_To value and add a ISOK_RESULTS.Resolved value without raising an error either update both values in the same **UPDATE** statement or first set the Deferred_To value to NULL and then the Resolved value to something non-NULL.

7.2.7 Resolved (Date and Time Resolved)

A timestamp value. Date and time the query result was resolved; that is, marked not a concern. The Isok system does not display resolved results, although of course the ISOK_RESULTS table can always be manually queried.

The value of this column may be `NULL`. This occurs both when the query result is a data integrity error and when it is a data integrity warning that has not yet been resolved.

7.2.8 Deferred_To

A timestamp value. Isok suppresses display of the result when the current time is before this time. Use of this column allows resolution of data integrity problems to be deferred, and hence not clutter up the output of Isok with noise that might hide other problems.

When this column is `NULL` Isok displays the query result.

7.2.9 Type

A `TEXT` value. Code classifying the query result. The legal values for this column are defined by the IR_TYPES support table.

This column may be `NULL` when the query result is unclassified.

7.2.10 Keep_Until

A timestamp value. This column controls whether or not run_isok_queries() deletes the row when the ISOK_QUERIES.Query is re-run and the query does not return the row's QR_ID.

A query result that the query no longer returns is kept until the given time is reached, when the value of this column is not `NULL`. When the value of this column is `NULL`, a query result row that is no longer returned is is always deleted. For further detail see the the section called "Deletion of Old Results" section of the run_isok_queries() documentation.

Tip
Using the special `TIMESTAMP` value of infinity entirely prevents deletion.

7.2.11 QR_ID (Query Result IDentifier)

A `TEXT` value. This is a unique, unique per query that is, identifier for the query result. It is the first column produced by the related ISOK_QUERIES.Query.

This column may not be `NULL`.

7.2.12 QR_Message (Query Result Message)

A `TEXT` value. This is the message, the second column, produced by the most recent execution of the ISOK_QUERIES.Query.

7.2.13 QR_Extra (Query Result Extra JSON data)

A `JSONB` value. The value of the third, optional, column returned by most recent execution of the query. This may contain any JSON deemed useful. This column serves as a catch-all container for any additional data that needs to be tracked regarding a reported condition.

The value of this column may be `NULL`. This is the default when the ISOK_QUERIES.Query does not return a third column.

See PostgreSQL's documentation on the JSON data types for information on how to access, index, and efficiently query the JSONB data type.

7.2.14 Notes

A `TEXT` value. Any notes regarding this particular query result. This column may not be `NULL`. This column may be empty; it need not contain characters, but it may not contain only whitespace characters.

8 The Support Tables

Support tables are used to control the values used in other tables. Each support table has a key, with an appropriate column name, and a column named `Description`. Both of these columns are of type `TEXT`. The keys of the support table are foreign keys of a column which has a controlled vocabulary, a limited number of terms which are allowed to be used.

An administrator can add or remove rows from the support tables to dynamically control the allowed vocabulary.

The support table `Description` columns must be unique when the comparison is made in a case-insensitive manner.

8.1 IQ_TYPES (Integrity Query Types)

IQ_TYPES contains one row for every code used to classify database integrity queries. Classification may be by the type of data integrity problem the related queries are designed to uncover, by who is responsible for resolving the discovered problems, or any other desired classification scheme.

8.1.1 Key: IQType

The IQ_TYPES table is keyed by the IQType column. This column may not contain whitespace characters. This column must be unique when compared in a case-insensitive fashion.

8.2 IR_TYPES (Isok Result Types)

IR_TYPES contains one row for every code used to classify or explain sets of database integrity problems, problems discovered by Isok's queries. Codes may be used as needed, whether to organize reported problems pending resolution, to describe the circumstances which resolve an issue, or to serve other purposes.

8.2.1 Key: IRType

The IR_TYPES table is keyed by the IRType column. This column may not contain whitespace characters. This column must be unique when compared in a case-insensitive fashion.

9 The Functions (Running Isok)

Isok is run by using one of its functions. Of course the ISOK_RESULTS table may always be queried manually, but this does not discover any new problems.

All of the Isok functions are designed to be used in the **FROM** clause of **SELECT** statements, as if they were tables. Indeed, the functions look like tables to the **SELECT** statement, tables that look exactly like ISOK_RESULTS -- except that the Resolved column is missing. The difference between querying on the ISOK_RESULTS table directly and querying using Isok's functions is that the functions update the content of the ISOK_RESULTS table by executing the the queries in ISOK_QUERIES table. Also, the functions never return rows where the underlying ISOK_RESULTS row has a non-`NULL` Resolved value or a Deferred_To time and date that has not yet been reached.

All timestamps, date plus time values, which Isok updates in the ISOK_QUERIES and ISOK_RESULTS tables are set to the date and time at which program execution started. So when, say, run_isok_queries(), is run, all of the new timestamp values in the ISOK_QUERIES and ISOK_RESULTS rows touched by the execution are identical.

Various Isok functions (or versions of the same function) are supplied to allow easy selection of which queries in which ISOK_QUERIES rows are to be executed, whether all or only some.

Note
As with a regular table, the order in which rows are returned by Isok's functions is unspecified. If you wish to ensure a specific ordering an **ORDER BY** clause must be used.

9.1 run_isok_queries

run_isok_queries — execute one or more of the queries stored in the ISOK_QUERIES table

Synopsis

TABLE (irid, iqname, first_seen, last_seen, type, qr_id, qr_message, notes) **run_isok_queries** (void);
TABLE (irid, iqname, first_seen, last_seen, type, qr_id, qr_message, notes) **run_isok_queries** (TEXT iqname_query);

Input

iqname_query

The text of an SQL query. The query must return a single column of ISOK_QUERIES.IQName values.

Description

A function which runs the queries stored in the ISOK_QUERIES table, returns the output of the stored queries -- excepting resolved and deferred rows, and stores the results in the ISOK_RESULTS table. Because the function returns rows and columns it is expected be invoked in the **FROM** clause of a **SELECT** statement. (See the Examples below.)

The function may be called in one of two ways. When called with no arguments all of the queries in ISOK_QUERIES are run. When called with the text of an SQL query, a query which returns a single column containing ISOK_QUERIES.IQName values, the function runs only those queries.

Tip
Use PostgreSQL's dollar quoting when supplying a query to **run_isok_queries()**.

The function returns a set of columns with multiple rows, a table. So it is expected to be used in the **FROM** clause of a **SELECT** statement. The columns returned by the function are the columns of the ISOK_RESULTS table, excepting the Resolved column.

The rows returned by the function are those returned by the queries the function executes, excepting resolved and deferred rows. A row is resolved or deferred when there is a row in ISOK_RESULTS that has a IQName value matching that of the query and a QR_ID value matching that of the row, and that ISOK_RESULTS row has a non-`NULL` Resolved column or a Deferred_To value that is in the future.

So, when called with no arguments the function returns all warning conditions that currently exist in the data, that have not been resolved or deferred, and all error conditions that currently exist in the data. When called with a query that selects specific ISOK_QUERIES to execute, only the unresolved, undeferred, warnings and errors discovered by the executed ISOK_QUERIES are returned.

Query Execution Order

When **run_isok_queries()** is called with no arguments, the queries are run in ISOK_QUERIES.IQName order, sorted lexically. When called with the text of an SQL query, the function runs the queries with the produced ISOK_QUERIES.IQNames, in the order given.

The Record of Query Execution

Running an ISOK_QUERIES.Query does more than add new rows to the ISOK_RESULTS table. Updates are made to existing rows to record and track the query execution's results.

The ISOK_QUERIES.Last_Run value is updated.

On ISOK_RESULTS, the rows to update are found by matching the ISOK_RESULTS.IQName value with the ISOK_QUERIES.IQName of the executed query, while also matching the QR_ID value with the value returned in the first column of the executed query. The columns updated are: Last_Seen , Last_Role , Last_Schemas , QR_Message , and QR_Extra.

Because the record of the results produced by Isok queries are updated, a query may be refined over time to produce enough information to resolve the reported issues.

Even though the execution of **run_isok_queries()** does not return rows that are resolved, all rows returned by an executed query have all the aforementioned columns updated to new values. Whether a row is returned or not does not matter, the update occurs anyway.

Deletion of Old Results

If an existing ISOK_RESULTS row matches the IQName value of the executed query, and there is no corresponding QR_ID value returned by the executed query, and the value of ISOK_RESULTS.Keep_Until is either `NULL` or **CURRENT_TIMESTAMP**[8] is not earlier than ISOK_RESULTS.Keep_Until, then the ISOK_RESULTS row is deleted. This empties the ISOK_RESULTS table of errors and warnings that no longer apply to the current state of the database.

If the query returns warnings, this deletion behavior does not depend upon whether or not the warning is resolved.

Examples

The following example runs all the queries in ISOK_QUERIES, displays all the errors and all the unresolved, undeferred warnings, ordered first by the name of the query, within that showing newer problems first, and within that ordered by warning id.

[8] The time the current transaction started, which, if a transaction was not explicitly started, is the time the database engine received the current SQL statement from the client and began execution.

Example 9.1 Executing all ISOK_QUERIES

```
SELECT *
  FROM run_isok_queries() AS problems
  ORDER BY problems.iqname
         , problems.first_seen DESC
         , problems.qr_id;
```

The following example runs a single saved query with an ISOK_QUERIES.IQName of `mycheck` and displays any of these sorts of problems found, ordered as in the previous example. This example also demonstrates how to use dollar quoting when supplying a a query as an argument to **run_isok_queries**, which thereby avoids problems having to do with trying to nest regular quotes.

Example 9.2 Executing a single ISOK_QUERIES.Query

```
SELECT *
  FROM run_isok_queries($$SELECT 'mycheck'$$) AS problems
  ORDER BY problems.iqname
         , problems.first_seen DESC
         , problems.qr_id;
```

The following example runs multiple specific queries, those with an ISOK_QUERIES.IQName of `mycheck`, `yourcheck`, and `theircheck`, and displays any of these sorts of problems found, ordered as in the previous example. As before, dollar quoting is used to quote the query which produces the IQNames.

Example 9.3 Executing many specific ISOK_QUERIES.Query-s

```
SELECT *
  FROM run_isok_queries($$VALUES ('mycheck')
                               , ('yourcheck')
                               , ('theircheck')
                            ORDER BY 1$$) AS problems
  ORDER BY problems.iqname
         , problems.first_seen DESC
         , problems.qr_id;
```

Notice that the query used to select the queries to execute has an `ORDER BY` clause. Without such a clause, the order in which the queries are run, and so the ordering of the rows returned, is unspecified.

The following example runs all the queries of the **bdate** type and displays any of these sorts of problems found, ordered as in the previous example. Again, dollar quoting is used.

Example 9.4 Executing ISOK_QUERIES of the “bdate” type

```
SELECT *
  FROM run_isok_queries(
         $$SELECT isok_queries.iqname
             FROM isok_queries
             WHERE isok_queries.type = 'bdate'
             ORDER BY isok_queries.iqname$$
       ) AS problems
  ORDER BY problems.iqname
         , problems.first_seen DESC
         , problems.qr_id;
```

A Security Considerations

The security concerns surrounding Isok are many, and can be complex. Fundamentally, this is because Isok executes arbitrary SQL. If the wrong SQL is executed, in the wrong context, anything might happen to your data. This appendix identifies pertinent issues, and how to minimize risk.

Ultimately, these are the same issues that arise in any application that executes SQL. The big difference between Isok and other applications is that most applications execute a more-or-less limited number of SQL queries that are carefully crafted to suit a specific purpose. The queries executed by Isok can have much more variation, and be subject to less review.

In the end, the recommendations here come down to following generally accepted security best-practices, in particular, the principle of least privilege.

A.1 Limiting Access

Limiting access to Isok is a clear first-step. Installing Isok into a dedicated `schema` goes a long way toward helping with this. When a schema is created, only the owner has access.[9] So, unless **GRANT**s are issued, access is limited by default.

Remember also, the ISOK_RESULTS table contains query output that may contain sensitive information to which access should be restricted. And, even if this is not true today, it may become true when additional queries are added to ISOK_QUERIES.

Even the queries in ISOK_QUERIES could, possibly, contain sensitive information.

[9] Yes, this is true of all objects. Only the owner has access to any newly-created object. But having a single point of access, the schema dedicated to Isok, that grants access to all of Isok, provides a very useful point of control that serves as an easily audited gateway to Isok’s functionality.

A.2 What Queries Access Matters

The executed queries, the ISOK_QUERIES.Querys, can be any SQL statement. Obviously, what executes matters. Less obviously, the ownership of and permissions granted on every object referenced by every query also matters.

Really, when multiple schemas are in the **search_path**, it is the ownership of and permissions granted on every object that *might be* referenced by every query that matters.

The ownership and permissions of referenced objects matter because these factors ultimately control what any given query actually does. If a user has, for example, permission to alter a view with some given name, or replace a table having that name with a view that has the same name, then the user can change what happens when that name is used in a query. The user can write a view that does anything. Or at least anything that the role which runs run_isok_queries() is allowed to do.

Imagine, the new view could call a function, say, in place of a table that was referenced, and that function could do anything at all. Even while still returning the replaced table's rows, so as to produce a results identical to that produced before the system was altered.

That is the issue. The user executing the saved query is dependent upon the goodwill of all the users who have enough access to alter any of the objects involved when the query is executed.

A.3 The Search Path

The ISOK_QUERIES.Search_Path column allows setting of the **search_path** on a per-query basis. The security implications of changing the search_path may be the hardest to reason through. The crux of the problem is that different users may have different permissions on the search path's schemas, and on the objects the schemas contain. This opens up the possibility that a malicious user may create an object, say, a view or a function, in a schema which appears earlier in the search path than the schema holding the object the query expects to find. If this is the case, the query will use the malicious object instead of the expected object.

The PostgreSQL documentation contains an analysis of this situation, in the context of writing **SECURITY DEFINER** functions. However, the analysis in the PostgreSQL documentation is not entirely applicable to Isok. In the case of Isok, even when Isok changes the effective role, the position in the search path of the temporary table schema, `pg_temp`, is less relevant. Because temporary tables are not shared between connections, the creation of a malicious object in the temporary schema must be done in the current connection. And so the issue is no different from that which occurs when any other malicious object is created in the current connection. In either case, there is a security lapse that occurs dynamically, at some point in the current connection.

Having said that, moving `pg_temp` to the end of the search path does make it harder to "mask" an existing object with a malicious object. Because all roles have permission to create objects in `pg_temp`, a malicious actor would not be able to mask an existing object with an object in `pg_temp` if `pg_temp` is at the end of the search path. For this reason it may make sense to always put `pg_temp` at the end of the search path whenever Isok is used.

The PostgreSQL documentation's observation remains valid: Malicious users with the ability to change objects in the search path may inject malicious objects.

A.4 Roles

The role in effect does have security implications. But changing a role for the duration of a query's execution, with ISOK_QUERIES.Role, has fewer security implications than it might seem.

Changing the current role does open up the possibility that database objects to which the new role has access may be changed. But this door is already open. A new role cannot be assumed without some chain of `SET` option grants from the `session_user` [definition here(-ish)] to the current role. So a malicious actor always has access to the same set of roles, regardless of whether Isok is involved or not.

What might be surprising is that, even though a role may **SET ROLE** to another, perhaps with less privileges, it is always possible to use **RESET ROLE** (or **SET ROLE NONE**) and reset the current role to the `session_role`. There is no sandboxing. If the session sets a role before running run_isok_queries(), there is the possibility that a malicious actor might undo the assumption of the role. This could then affect the role used to execute any queries that run_isok_queries() has not yet executed.

Don't expect that a **SET ROLE** to a role of lesser privileges makes running run_isok_queries() any safer.

A.5 Mitigation Strategies

There is no one-size-fits-all solution. Even disabling Isok's ability to dynamically alter the current search path and the current role does not address the fundamental issues. Even more so because, to be useful, run_isok_queries() may need an expansive set of permissions to do its job.

One possible strategy is to always supply values in the ISOK_QUERIES.Role ISOK_QUERIES.Search_Path columns. At least that way the context of each query's execution is always known.

Another possible strategy is to install Isok in multiple schemas, each schema dedicated to a different purpose and assigned different permissions, intended to be used by different users.

A.6 Creating an Audit Trail

To better respond to a suspected security problem it is always very useful to have an audit trail to examine. One way to have such a trail is to install a temporal extension. These extensions track the history of database content over time. The Isok tables could be temporally tracked, to audit what queries were changed when, as well as what query results were produced or deleted when.

A conceivable, although entirely untested on our part, idea is to use a temporal extension to track changes made to the `postgres` database. Otherwise known as the system catalog, `pg_catalog`, this database contains the definitions of all objects in all databases. Tracking the catalog provides an audit trail should a malicious object be created, although this would not help if `pg_temp` was involved.

Some installations may even want to temporally track all their tables, although this may not be feasible for a whole host of reasons.

B Some Notes on Query ID Style

The first column returned by an ISOK_QUERIES.Query, the ID column, must return a unique value. That is, unique within the set of rows produced by the query.

There are two possible approaches to take when deciding what value to return in the query's ID column. The first is to take a minimalist approach and use the shortest possible unique value. This has the advantage of being easy to type when manually writing SQL that contains the ID.

This is the approach taken by the example which describes testing for customers that have no orders, reproduced here:

```
SELECT customers.id
     , 'Customer ' || customers.id || ' has no related ORDERS'
     , NULL
  FROM customers
  WHERE NOT EXISTS (SELECT 1
                      FROM orders
                      WHERE orders.customerid = customers.id);
```

An alternate approach is to make the ID column descriptive. The advantage of this approach is that it provides a summary of the problem in the ID column, so that the second column the query returns, the Msg column, need not be examined when cursorily looking over Isok's output. This can be important if the second column, the Msg column, contains a variety of additional information that might be useful in resolving the issue.

Here is the test for customers without orders, re-written so that the ID column summarizes the problem and the Msg column provides useful detail:

```
SELECT 'Customer ' || customers.id || ' has no related ORDERS'
     , 'The problem customer is: Key (ID) = ' || customers.id
       || '), Value (Name) = (' || customers.name
       || '), Value (Country) = (' || customers.country
       || '), Value (State) = (' || customers.state
       || ')'
     , NULL
  FROM customers
  WHERE NOT EXISTS (SELECT 1
                      FROM orders
                      WHERE orders.customerid = customers.id);
```

C Frequently Asked Questions

Q: *Where do I get support?*

A: File an issue at Codeberg.

Q: *How do I find out about new pg_isok releases?*

A: Announcements of new pg_isok releases are made on the PostgreSQL announcement email list. Subscriptions are managed at lists.postgresql.org.

Q: *I'm installing a cloud "pure SQL" variant and I get a lot of errors, beginning with one containing* `ERROR: function isok_queries_update_func() does not exist`.

A: You ran **make TARGET_SCHEMA=myschema ...**, but either `myschema` does not exist or you do not have adequate permissions.

D Local Copies of the Documentation

When Isok is installed as an extension, local copies of the documentation are installed. The pg_config PostgreSQL client command provides an easy way to find the documentation.

Example D.1 Finding the Documentation of Locally Installed Extensions

```
$ printf '\nExtension documentation is located in:\n%s\n\n' $(pg_config -- ↩
    docdir)/extension/

Extension documentation is located in:
/usr/share/doc/postgresql-doc-15/extension/

$ ls $(pg_config --docdir)/extension
pg_isok--1.0.0.config  pg_isok_html  pg_isok_usletter.pdf
pg_isok_a4.pdf         pg_isok.txt

$ printf '\nThe URL used to read the local HTML documentation is:\nfile ↩
    ://%s\n\n' \
          $(pg_config --docdir)/extension/pg_isok_html/html_paginated/index ↩
              .html

The URL used to read the local HTML documentation is:
file:///usr/share/doc/postgresql-doc-15/extension/pg_isok_html/ ↩
    html_paginated/index.html
```

E Periodic Execution

A monitoring system must periodically execute and deliver reports if it is to monitor and provide actionable alerts on an ongoing basis. Isok does not include a periodic job scheduler. Tools like the Unix cron

command, the systemd timer system, or the PostgreSQL pg_cron extension are useful to automate, and make periodic, Isok's monitoring. There are plenty of job schedulers available and one of these must be used to schedule the production of Isok's reports.

Typically, something must deliver the reports Isok produces, because push-notifications remind people to act. Although Isok does archive the reports it produces, it does not include a report delivery mechanism. Email, or other push-based delivery mechanisms (perhaps email-to-SMS text gateways), are the expected delivery mechanisms for Isok's reports. Isok itself can report to standard out when run from **psql**. Depending on your job scheduler, some amount of scripting may be required to route Isok's reports to a push delivery service.

E.1 Example Periodic Reporting via Email Using systemd

The files shown below deliver an Isok report, if there is something to report, by email every Tuesday morning.

The system on which they are installed must have a mail transfer agent installed, like `Postfix`, to begin the email delivery process. The system also must have GNU `mailutils` installed, or an equivalent **mail** command, like BSD **mailx**, to send the email.

Most operating systems will have packages available to install these services, and a way to configure simple defaults. However, it is non-trivial to reliably deliver email from your system directly to the rest of the Internet. The recommended approach is to send the email from your local system to a *mail relay* provided by your local IT professionals. (Or, your Internet Service Provider. Or, if you are hosted in the cloud, your hosting company.) These professionals will usually be able to supply you with what you need to know to have mail sent from your system to a system able to send email to the Internet at-large. If not, there are companies that provide this service for a nominal fee.

The service you would ask for is usually called *an email relay service*.

It is usually a good idea to ask your local IT professionals to help with the selection of a mail transfer agent.

This example is expected to run, as is, on most systems that have the default PostgreSQL install.

The example connects to the database and runs as the `postgres` role, the bootstrap superuser. It assumes that `pg_hba.conf` contains:

```
local   all             postgres                                ↩
    peer
```

This line is typically present, but this is not guaranteed.

Note
When cutting and pasting from the examples, don't forget to remove the "callout" numbers -- the numbers that call attention to particular lines and have annotations below. Leaving them in can result in errors that are difficult to debug.

Example E.1 Sample `/etc/systemd/system/isok_report.service` File

```
# This file is: /etc/systemd/system/isok_report.service ❶
[Unit]
Description=Run pg_isok's run_isok_queries() function and email when there ↩
    's \
a result
ConditionACPower=true

[Service]
#
# Configuration is done here (and in /etc/aliases, see pg_isok_report)
#

# The postgres connection string (or other arguments to psql)
# Putting passwords in here is a bad idea, change pg_hba.conf instead?
Environment="CONNECTION_STRING=mydatabase" ❷

# Put a connection string variable assignment containing secrets in this  ↩
    file:
# (man 5 systemd.exec)
#EnvironmentFile=/etc/pg_isok_secrets ❸❹

# The schema in which pg_isok is installed
Environment="ISOK_SCHEMA=isok" ❺

# End of configuration

# The Uni*x user running the db engine
# (Expected to be the same as the PostgreSQL bootstrap superuser)
User=postgres ❻
Type=oneshot
KillMode=process

PassEnvironment=CONNECTION_STRING ISOK_SCHEMA
ExecStart=/usr/local/bin/pg_isok_report
```

❶ After installation, or change in the content, don't forget to run:

```
systemctl daemon-reload
```

❷, ❸, ❺ Configuration settings

❹ Do not forget to set appropriate permissions on the secrets file.

❻ This is the Un*x `postgres` user, which is usually has the same name as the PostgreSQL bootstrap superuser. So the supplied connection parameters don't mention the username because the default is to use a role with the same name as the connecting Un*x user.

Example E.2 Sample `/etc/systemd/system/isok_report.timer` File

```
# This file is: /etc/systemd/system/isok_report.timer ❶
[Unit]
Description=Tuesday report from pg_isok

[Timer]
# See: man 7 systemd.time
OnCalendar=tuesday *-*-* 3:00 ❷
RandomizedDelaySec=60m ❸
Persistent=true

[Install]
WantedBy=timers.target
```

❶ After installation, or change in the content, don't forget to run:

```
systemctl daemon-reload
systemctl enable pg_isok_report.timer
systemctl start pg_isok_report.timer
```

❷, ❸ Configuration settings

Example E.3 Sample `/usr/local/bin/pg_isok_report` File

```
#!/usr/bin/bash
# This file is: /usr/local/bin/pg_isok_report ❶
#
# Run pg_isok, and mail (with GNU mailutils) if it produces anything.
#
# Expected environment variables:
# CONNECTION_STRING
#    The postgres connection string (or other arguments)
#    Putting passwords in here is a bad idea, change pg_hba.conf instead?
# ISOK_SCHEMA
#    The schema in which pg_isok is installed

# The recommendation is to _not_ change this.  Instead, make an
# alias for "pg_isok_report" in /etc/aliases.
MAIL_RECIPIENT=pg_isok_report

EMPTY_FILE=$(/usr/bin/mktemp --tmpdir pg_isok_empty.XXXXXXXXXX)
OUTPUT=$(/usr/bin/mktemp --tmpdir pg_isok_output.XXXXXXXXXX)
```

```
PSQL="/usr/bin/psql ${CONNECTION_STRING}"

cleanup () {
  /usr/bin/rm -rf ${EMPTY_FILE} ${OUTPUT}
}
trap cleanup EXIT

PAGER= ${PSQL} --command="
              SELECT irid, iqname, first_seen, last_seen, last_role
                   , last_schemas, deferred_to, type, keep_until
                   , qr_id, qr_message, qr_extra, notes
                FROM ${ISOK_SCHEMA}.isok_results
                LIMIT 0;
  " \
  > ${EMPTY_FILE} 2>&1

PAGER= ${PSQL} --command="SELECT * FROM ${ISOK_SCHEMA}.run_isok_queries() ↩
    ;" \
  > ${OUTPUT} 2>&1

cmp --quiet ${EMPTY_FILE} ${OUTPUT} \
  || { /usr/bin/mail -s 'Isok output' ${MAIL_RECIPIENT} \
         < ${OUTPUT} ; }
```

❶ After installation, don't forget to run:

```
chmod a+x /usr/local/bin/pg_isok_report
```

F Techniques For Making Local Extensions to Isok

Should you find yourself wishing that Isok did more, here are some suggested techniques for extending the functionality of your Isok instance. There is overlap, more than one technique may facilitate reaching any given goal.

F.1 Wrap run_isok_queries()

To perform actions before or after execution of run_isok_queries(), write a new function that takes run_isok_queries()'s arguments and returns run_isok_queries()'s results. And does what you wish before or afterward.

So, for example, to ensure a safe, consistent, value for **search_path**, you could write a function that executes **SET search_path ...;**, before itself calling run_isok_queries() and returning the result.

F.2 Extend Issue Classification

If you would like additional ways to classify the issues your queries discover, the IR_TYPES table may be extended.

Create your own table to do this, called, say, **IR_TYPE_CLASSES**.

The key of this table is that of the IR_TYPES table; may as well call it **IRType**. It is a foreign key, referencing IR_TYPES. So your new table has a one-to-one relationship with IR_TYPES.

Add as many columns as you like to your new table, a column for each (orthogonal) sub-category by which you would like to classify reported issues. Boolean columns behave as a tag, toggling classification. Other kinds of columns, possibly containing foreign keys to control the vocabulary used, allow richer classification schemes.

F.3 Fully Utilize ISOK_RESULTS.QR_Extra

Indexing the JSONB ISOK_RESULTS.QR_Extra column improves performance.

If you know your JSONB keys, you can make a VIEW that exposes the value of those keys as the view's columns. Users of this view would not have to be familiar with querying JSONB.

More complex schemes involve putting a row-level **BEFORE** trigger on ISOK_RESULTS to distribute the various values appearing in ISOK_RESULTS into other tables. But doing so surely takes you past the point of diminishing returns. It is easier to modify the SQL that Isok installs. And doing so is probably also less of a long-term maintenance burden, which matters.

F.4 Modify Isok's Generated SQL

Should you want to make a modification like allowing the queries in ISOK_QUERIES.Query to return additional columns, you can do so by modifying the SQL that Isok loads.

Isok is pure SQL, so the SQL can be generated and then modified, in any way you like, before being loaded into a database's schema.

G Developing

We consider Isok to be feature complete. That said, there's always room for improvement and contributions are welcome. Never the less, if you would like your changes added to Isok, before doing a lot of work we recommend communicating with us.

You are, of course, free to make changes to your local Isok.

Development should be done by cloning the git repository.

G.1 Tool Requirements

Isok uses the PGXN.org tools for building and distribution, which in turn uses parts of the PostgreSQL GNU make-based build system. So GNU make is required. In addition, the Isok documentation is done with the DocBook, as is PostgreSQL's, so the tooling required to build the documentation is the same as PostgreSQL, although Isok currently generates XHTML so may require a slightly different set of DTDs.

Aside from these requirements, the following additional tools are needed:

DBLatex The DocBook to LaTeX to PDF, etc., converter

gawk The GNU awk implementation

Gnu m4 The macro pre-processor used by GNU autoconf, etc.

links The command-line web browser

xmllint The XML linter

zip The archive and compression tool

DBLatex also requires the installation of various TeX and LaTeX tooling, which your O/S's package manager is likely to install as a dependency.

When working with DocBook, the book DocBook XSL: The Complete Guide from Sagehill.net may also be useful.

G.2 Building and Distributing

Run **make help** for help on the Makefile targets.

Almost all the generated files are included in the distribution. This is so that the user, or the PGXN tooling, can use the Makefile for installation, and uninstallation, without having to have all the tooling required for development installed.

H Acknowledgments and History

Isok was first developed as "The Warning System" for the Gombe Mother Infant Database Project. It was later incorporated into Babase, part of The Amboseli Baboon Research Project, and enhanced to take advantage of the features in PostgreSQL 9.1. Further enhancement, including release as a PostgreSQL extension, was done for The SokweDB Project, developed by The Jane Goodall Institute.

We would like to thank these projects, and their funding sources, for enabling the development and release of Isok.

The acknowledgments included in the above projects' documentation are reproduced below. (Verbatim, excepting some updated contact information.) It is not clear how applicable the entirety of the acknowledgments are, but we would rather be overly generous in our thanks than be stingy.

The following acknowledgments do not include all the people who have enabled and assisted Isok development. You know who you are. Thank you. And thanks to the larger Open Source community. Without their support, and hard work, none of this would have happened.

H.1 The Gombe Mother-Infant Project Acknowledgments

THE GOMBE-MI DEVELOPMENT GROUP

Karl O. Pinc Book Author, System Design Lead, Implementation

PhD.. Carson M. Murray Project Co-Leader, System Design Core Member

PhD.. Elizabeth V. Lonsdorf Project Co-Leader, System Design Core Member

Karen Anderson System Design Core Member, Copy Review, System Testing Lead

PhD.. A. Catherine Markham System Design Participant

PhD.. Margaret A. Stanton System Design Participant, System Testing Core Member

Jr.. Edward Wilkerson System Design Participant

Funding and Support We gratefully acknowledge the support of The National Institutes of Health grant R00HD057992 for the development of this system. We are also very grateful for the support given by The Leo S. Guthman Foundation, the Lincoln Park Zoo, Franklin & Marshall College, and The George Washington University.

Other Thanks We would like to thank the myriad Free and Open Source communities, including those of PostgreSQL, the GNU Project, the Debian Project, Ubuntu, PhpPgAdmin, the Pyramid web framework, TeX and LaTeX, DBLatex, DocBook, Babase, and many others unmentioned, for giving, gratis, billions of dollars[10] of work to the world, without which the Gombe-MI software and this book would not exist.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Institutes of Health, The Leo S. Guthman Foundation, the Lincoln Park Zoo, Franklin & Marshall College, The George Washington University, or any other organization which has supplied support for this work.

[10] See: Estimating the Total Development Cost of a Linux Distribution.

H.2 The Babase Acknowledgments

We gratefully acknowledge the support of the National Science Foundation for the supporting the collection of the majority of the data stored in the database; in the past decade in particular we acknowledge support from IBN 9985910, IBN 0322613, IBN 0322781, BCS 0323553, BCS 0323596, DEB 0846286, DEB 0846532 and DEB 0919200. We are also very grateful for support from the National Institute of Aging (R01AG034513-01 and P01AG031719) and the Princeton Center for the Demography of Aging (P30AG024361). We also thank the Chicago Zoological Society, the Max Planck Institute for Demographic Research, the L.S.B. Leakey Foundation and the National Geographic Society for support at various times over the years. In addition, we thank the National Institute of Aging (R03-AG045459-01) for supporting recent work extending the database to incorporate genetic and genomic data.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the National Institute of Aging, the Princeton Center for the Demography of Aging, the Chicago Zoological Society, the Max Planck Institute for Demographic Research, the L.S.B. Leakey Foundation, the National Geographic Society, or any other organization which has supplied support for this work.

H.3 The SokweDB Acknowledgments

At the time of this writing, there is no formal set of acknowledgments for SokweDB.

However, Microsoft provided funding for SokweDB and we would like to acknowledge and thank them for their support.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Microsoft.

I Licensing Terms -- Licensed Under The AGPL v3.0+ (Examples Excepted)

J GNU Affero General Public License version 3

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU Affero General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major

Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its

resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own

removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your

license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published

by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<i>one line to give the program's name and a brief idea of what it does.</ ↩
    i>
Copyright (C) <i>year</i> <i>name of author</i>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published  ↩
    by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see https://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see https://www.gnu.org/licenses/.

K CC0 1.0 Universal Deed

K.1 No Copyright

The person who associated a work with this deed has dedicated the work to the public domain by waiving all of his or her rights to the work worldwide under copyright law, including all related and neighboring rights, to the extent allowed by law.

You can copy, modify, distribute and perform the work, even for commercial purposes, all without asking permission. See Other Information below.

K.2 Other Information

In no way are the patent or trademark rights of any person affected by CC0, nor are the rights that other persons may have in the work or in how the work is used, such as publicity or privacy rights.[11]

Unless expressly stated otherwise, the person who associated a work with this deed makes no warranties about the work, and disclaims liability for all uses of the work, to the fullest extent permitted by applicable law.

When using or citing the work, you should not imply endorsement by the author or the affirmer.[12]

K.3 Notice

The Commons Deed is not a legal instrument. It is simply a handy reference for understanding the CC0 Legal Code, a human-readable expression of some of its key terms. Think of it as the user-friendly interface to the CC0 Legal Code beneath. This Deed itself has no legal value, and its contents do not appear in CC0.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this Commons Deed does not create an attorney-client relationship.

Creative Commons has not verified the copyright status of any work to which CC0 has been applied. CC makes no warranties about any work or its copyright status in any jurisdiction, and disclaims all liability for all uses of any work.

[11] *publicity or privacy* — The use of a work free of known copyright restrictions may be otherwise regulated or limited. The work or its use may be subject to personal data protection laws, publicity, image, or privacy rights that allow a person to control how their voice, image or likeness is used, or other restrictions or limitations under applicable law.

[12] *endorsement* — In some jurisdictions, wrongfully implying that an author, publisher or anyone else endorses your use of a work may be unlawful.

L CC0 1.0 Universal