Overview

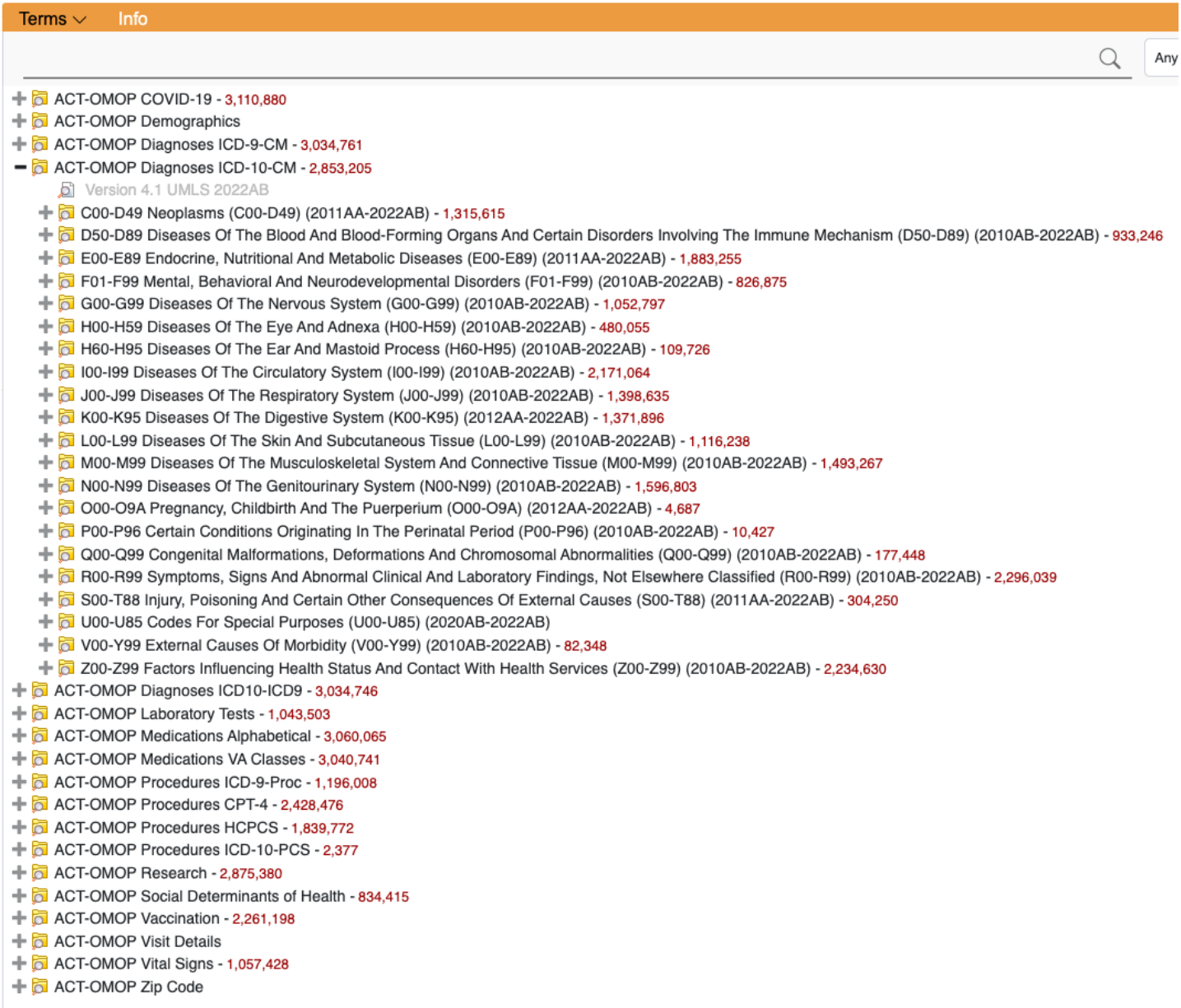
Patient Counting Scripts ("totalnum") · https://i2b2.org/docs/patient-counting-scripts-totalnum/

i2b2 ontologies have a column c_totalnum that can store the total count of patients associated with every item in the ontology tree. This can be visualized in the i2b2 webclient to assist with query building (e.g., to find concepts that have many patients) or be used for data quality (to find areas where patient counts do not make sense). It is also used by the query builder to optimize queries. The ENACT network uses these counts for additional analytics across their network. We recommend you run these counts after each ETL.

i2b2 users must have the DATA_AGG user permission to view the counts through the web client.

The stored procedures loaded in the Metadata schema must have read access to the CRC schema (more information in Installation below).

Mapping codes through the concept_dimension or Adapter Mappings files are not supported. i.e. the c_basecodes in your ontology tables must be the same codes used in your fact tables.



Installation

i2b2 includes the totalnum stored procedures and they are installed as part of the data installation. To manually install/upgrade, follow these steps:

1. Download the i2b2 upgrade package from i2b2.org, or clone the Github repo.
2. Navigate to the data package, under Release_1-8/Upgrade/Metadata and edit the db.properties file as needed.

3. If your i2b2 is below version 1.7.12a, run the ant upgrade script to create the totalnum and totalnum_report tables. See the upgrade guide for complete information.
   e.g., ant -f data_build.xml upgrade_tables_release_1-7-12a
4. Run the ant script target to create the stored procedures.
   ant -f data_build.xml rebuild_procedures
5. Set privileges: If using multiple schemas, the stored procedure should be run from the *metadata* schema. Make sure the stored procedure can read the tables in the *crcdata* schema (observation_fact, visit_dimension, patient_dimension) and can both read an update ontology tables in the *metadata* schema (including table_access).

The code for the totalnum stored procedures can be found here.

Fast Totalnum

Got it--here's your updated doc with Oracle and Postgres run syntaxes included and formatted as quick, copy-paste blocks.

Fast Totalnum

i2b2 1.8 introduces a version that is 5-10× faster. As of 1.8.2, this faster version is available for MSSQL, Postgres, and Oracle. Counting the patient and visit dimension requires the ACT ontology. These replace the `pat_count_dimensions` and `run_all_counts` stored procedures.

Configuration

One-time / when ontology changes: run the preparatory routine. This creates:

- `OBSFACT_PAIRS` view (distinct `concept_cd, patient_num`)
- `TNUM_ONTOLOGY` (unified ontology table)
- `CONCEPT_CLOSURE` (transitive closure table)

This can take up to ~1 hour.

- If you use more than one fact table, customize the `obsfact_pairs` view (see code comments).
- `ACT_VISIT_DETAILS_V41` and `ACT_DEM_V41` table names are presently hardcoded--change if your local names differ.

Counting: runs with no parameters; writes to the `totalnum` table (created during i2b2 1.7.12/1.7.13/1.8 upgrade). Typical runtime: 1-3 hours.

Output: writes obfuscated results to `totalnum_report` and pushes `totalnum` into ontology tables for the UI.

Execution (by database)

SQL Server (T-SQL)

```
-- Run once (when ontology changes)
EXEC FastTotalnumPrep;                 -- or:
EXEC FastTotalnumPrep 'dbo';           -- specify schema

-- Actual counting (hours)
EXEC FastTotalnumCount;

-- Output results to report + ontology (all ontology tables)
EXEC FastTotalnumOutput;               -- or:
EXEC FastTotalnumOutput 'dbo','@';     -- schema + all ontologies

-- Single ontology table example
EXEC FastTotalnumOutput 'dbo','MY_ONTOLOGY';
```

Oracle

> Tip: In SQL*Plus/SQLcl, end each anonymous PL/SQL block with `/`. In SQL Developer, you can run the block directly.

```
-- Run once (when ontology changes)
BEGIN
```

```
  FastTotalnumPrep('MY_SCHEMA');  -- optional schema
END;
/

-- Actual counting (hours)
BEGIN
  FastTotalnumCount;
END;
/

-- Output results (all ontology tables = default)
BEGIN
  FastTotalnumOutput;
END;
/

-- Output results (single ontology table)
BEGIN
  FastTotalnumOutput('MY_SCHEMA', 'MY_ONTOLOGY');
END;
/
```

Postgres

> Note: Some installs expose `fasttotalnumprep` as a function (call with `SELECT`) and `fasttotalnumcount/fasttotalnumoutput` as stored procedures (call with `CALL`). Replace the schema (`public` below) with yours--often `public`.

```
-- Run once (when ontology changes) on default schema
SELECT fasttotalnumprep();

-- Or on a specific schema
SELECT fasttotalnumprep('my_schema');

-- Actual counting (hours)
CALL fasttotalnumcount();

-- Output results (all ontology tables)
```

```
CALL fasttotalnumoutput();

-- Output results (single ontology table)
CALL fasttotalnumoutput('public', 'my_ontology');
```

Some additional notes on running on OMOP

It is possible to run counts on OMOP tables through the ENACT-OMOP feature in i2b2 1.8. The new 1.8 totalnum procedure works on OMOP - simply load the file `totalnum_usp/sqlserver/totalnum_fast_prep_OMOP.sql` instead of `totalnum_fast_prep.sql`.

Totalnum Classic (slower but more compatible with ontologies that use dimcode-based queries)

Configuration

1. If using multiple fact tables, the recommended approach is to create a fact table view as the union of all your fact tables. (This is essentially going back to a single fact table, but it is only used for totalnum counting. This is needed to correctly count patients that mention multiple fact tables within a hierarchy.)

   ```
   e.g.,
      create view observation_fact_view as
      select * from CONDITION_VIEW
      union all
      select * from drug_view
   ```

   1. If running the counting script in SQL Server, add the wildcard flag, to ignore multifact references in the ontology: e.g. exec RunTotalnum 'observation_fact_view','dbo','@','Y'
      This is automatically accounted for in the other database platforms. Note this approach does not work if you have conflicting concept_cds across fact tables.

Execution

See database-specific instructions below. After running the scripts, results are placed in: c_totalnum column of all ontology tables, the totalnum table (keeps a historical record), and the totalnum_report table (most recent run, obfuscated). These total counts will also be visible in the ontology browser in the web client.

MSSQL Version

By Mike Mendis and Jeff Klann, PhD based on code by Griffin Weber, MD, PhD

Run with:

```
exec RunTotalnum or exec RunTotalnum 'observation_fact','dbo','@'
```

The optional parameters are:

1. Observation table name (for multi-fact-table setups)
2. Schema name
3. A single ontology table name (specify to to run on a single ontology table - otherwise (or if '@' is specified) runs on all tables in table_access)
4. A wildcard flag that will ignore multifact references in the ontology if 'Y'. (See below for the use case.)

Note that visit and patient dimension will only be counted in conjunction with the default (observation_fact) tablename!

To use with multi-fact-table setups:

Option 1) If you have at most one fact table per ontology, run this once with each fact table specified!
e.g., to use on a fact table called derived_fact with just the act_covid ontology: exec RunTotalnum 'derived_fact','dbo','act_covid'

Option 2) Create a fact table view as the union of all your fact tables. (This is essentially going back to a single fact table, but it is only used
for totalnum counting. This is needed to correctly count patients that mention multiple fact tables

within a hierarchy.)

e.g.,

Example 1: Counting using OMOP tables

```
create view observation_fact_view as
select * from CONDITION_VIEW
union all
select * from drug_view
```

And then run the totalnum counter with the wildcard flag, to ignore multifact references in the ontology, e.g.,

```
exec RunTotalnum 'observation_fact_view','dbo','@','Y'
```

Example 2: Counting using a derived fact table and the regular fact table, using a single ontology

```
create view observation_fact_view as
select * from observation_fact
union all
select * from derived_fact
```

Run the totalnum counter with the wildcard flag, to ignore multifact references in the ontology, and specify an ontology table, e.g.,

```
exec RunTotalnum 'observation_fact_view','dbo','act_covid_v4','Y'
```

Note this approach does not work if you have conflicting concept_cds across fact tables.

Oracle Version

By Mike Mendis, based on SQL Server code by Griffin Weber, MD, PhD

Performance improvements by Jeff Green and Jeff Klann, PhD 03-20

Run the procedure like this (but with your schema name instead of i2b2demodata):

```
begin
runtotalnum('observation_fact','i2b2demodata');
end;
```

You can optionally include a table named if you only want to count one ontology table (this IS case sensitive):

```
begin
runtotalnum('observation_fact','i2b2demodata','I2B2');
end;
```

Note: If you get the error as: ERROR at line 1: ORA-01031: insufficient privilege, then run the command:
grant create table to (DB USER)

Postgres Version

Original PostgreSQL code by Dan Vianello, Center for Biomedical Informatics, Washington University in St. Louis
2019 - Modified for i2b2 1.7.12 release by Mike Mendis, Partners Healthcare
2020 - Updated to support reporting and single-table runs by Jeff Klann, Massachusetts General Hospital

Usage example:

```
select runtotalnum('observation_fact','public')
```

- Replace 'public' by the schema name for the fact table.
- If using a schema other than public for metadata, you might need to run "set search_path to 'i2b2metadata','public' " first
- You can optionally specify a single table name, to count using only one ontology table. This is case sensitive.

Running using ANT

Run the ant command to execute the data_build.xml file with below specified target

- POSTGRESQL : ant -f data_build.xml db_metadata_run_total_count_postgresql
- ORACLE : ant -f data_build.xml db_metadata_run_total_count_oracle
- SQL SERVER : ant -f data_build.xml db_metadata_run_total_count_sqlserver

```
```

Some additional notes on running Postgres

Some users have reported difficulty executing the totalnum scripts due to user permissions. Lav Patel at UKMC has offered some solutions:

1. Make sure the i2b2 user has access to insert, select, and update all i2b2 schemas... e.g., `GRANT ALL PRIVILEGES ON DATABASE i2b2 to i2b2`
2. Make the i2b2 user a super user: `ALTER USER i2b2 with SUPERUSER;`
3. Change the schema ownership to the i2b2 user (requires function in the postgres directory of this repository):

```
select change_schema_owner('i2b2demodata', 'i2b2');
select change_schema_owner('i2b2metadata', 'i2b2');
select change_schema_owner('i2b2pm', 'i2b2');
select change_schema_owner('i2b2hive', 'i2b2');
```

Output

The scripts produce three outputs:

1. The c_totalnum columns in your ontology tables contain numbers (not nulls). These total counts will be visible in the ontology browser in the web client.
2. The totalnum table will be appended with a row for each concept that was counted, with its count and the date. This can be used for analytics on totalnum counts.
3. The totalnum_report will be truncated and populated with a row for each concept that was counted, with an obfuscated count using the same approach as SHRINE and the date. This can be

used for sharing deidentified counts with collaborators (with IRB approval). (Note that the date is the date the script was run, and so it is not a HIPAA-protected date.)

Parent folders will get counts (of all patients with facts in the leaves) *except* for ontology folders derived from visit_dimension or patient_dimension. These cannot be rolled up because of the way these terms are defined in the ontology. They will have no count at all (not a zero).