DATABASE SCHEMA FOR THE ICEWS RCDR

Jennifer Lautenschlager
(jlautens@atl.lmco.com)

November 14, 2013

Revision History

Date	Author	Description
11/14/2013	J. Lautenschlager	Pulled schema section out of the ISPAN Data Tool document and into its own document here, then augmented

TABLE OF CONTENTS

TABLE OF FIGURES 4
TABLE OF TABLES 4
1 Overview 5
2 Related Documents 5
3 Database Schema 5
3.1 Actor Dictionary Tables 7
3.1.1 Sectors 7
3.1.2 Actors vs Agents 10
3.1.3 Actor Hierarchies 14
3.2 Geographic Tables 16
3.2.1 Countries, COCOMS, and Regions 16
3.2.2 Gazetteer Tables 19
3.3 Story Related Tables 21
3.3.1 Publisher Tables 22
3.3.2 Other Story Related Tables 22
3.4 Event Related Tables 26
3.4.1 The Eventcoders Table 26
3.4.2 The Eventtypes Table 26
3.4.3 Events and Badevents Tables 27
3.4.4 Simple_events and Simple_badevents Tables 31
3.4.5 Serif_events and Serif_badevents Tables 32
3.4.6 Event Mapping Tables 32
3.5 Event Aggregation Tables 33
3.6 State Data Tables 34
3.7 Other Tables 36
3.7.1 Event Browser Tables 36
3.7.2 ISPAN Security Tables 37
3.7.3 Deprecated Tables 37
Appendix A: Glossary 38

TABLE OF FIGURES

Figure 1 - Sector-related Database Tables 9
Figure 2 - Actor-related Database Tables 13
Figure 3 - Actor Hierarchy-related Database Tables 15
Figure 4 – Countries-, COCOMs-, and Region-related Database Tables 18
Figure 5 – Gazeteer-related Database Tables 21
Figure 6 - Story-related Database Tables 25
Figure 7 - Event-related Database Tables 30
Figure 8 - Aggregation-related Database Tables 34
Figure 9 - State Data-related Database Tables 36

TABLE OF TABLES

Table 1 - ICEWS RCDR Tables 6
Table 2 – Sector-related Database Tables 9
Table 3 – Countries-, COCOMs-, and Region-related Database Tables 17
Table 4 – Gazetteer-related Database Tables 19
Table 5 – Story-related Tables 23
Table 6 - Stories Table Fields 24
Table 7 - Events and Badevents Tables Fields 28
Table 8 - Augmented simple_events and simple_badevents Tables Fields 32
Table 9 - Aggregation-related Database Tables 33

1 Overview

This document describes the database schema for the ICEWS Raven Core Data Repository (RCDR) used by the iTRACE component**.**

2 Related Documents

- Event Aggregations on ICEWS: Provides in-depth detail about aggregations (also known as aggregated event data) in ICEWS, focusing on related concepts, how aggregations are defined, how they are calculated, and nuances related to using aggregations in analytical work.
- ISPAN Data document: This technical document is geared toward the administrator who would be running the ICEWS Data Processing Server (DPS). It includes a high level view of the event coding process, detailed step-by-step explanations of the process, steps to take to QC processed data, suggested data ConOps, details on how to hand-edit the database, and various processing scripts related to maintaining the DPS.
- Dictionary Update (only slightly technical): An only slight technical description of how dictionary updates work, what they entail, and why they are so complex.
- Geolocation of Event Data on ICEWS: Provides a high-level technical overview of how geolocations are determined for ICEWS events.

3 Database Schema

The table below lists the various tables by type. Anything not found is considered to be a legacy table.[1]

iTRACE Core[2]	**iCAST Core**	**iTRACE DPS**	**iCAST DPS**
actorfilter	aggmodsumview	altnames	dataparam_events_mapping
actorfilter_countries	aggmodsumview_details	*badevents*	dataparameters
actorfilter_dict_sectors	aprview	*badstories*	datapoints
actorfilter_selections	aprview_aor_models	*bad_serif_events*	*datapoints_change_annotations*
cocom_actor_mappings	aprview_ctry_models	country_info	*datapoints_change_log*
cocom_regions	basedataparameters	countrynamechange	datasources
cocoms	classifications	dict_agent_sector_mappings	dict_etcons_et_mappings
countries	components	dict_agentpatterns	dict_seccons_sec_mappings
dict_actorgroups	dateranges	dict_agents	dict_eventqueries
dict_actgrp_dict_ actnodes	eoiganttevent	dict_sector_actor_mappings	dict_eventqueryconstraints
dict_actorgroups_ dict_actors	eoiganttevent_events	event_source_country_ mappings	dict_eventquery_methods
dict_actor_agent_ mappings	ganttchartview	event_target_country_ mappings	dict_eventquery_method_types
dict_actorlinks	ganttchartview_events	eventcoders	event_source_sector_mappings
dict_actornodes	ganttevent	*factiva_region_hierarchy*	event_target_sector_mappings
dict_actorpatterns	group_subset_parameters	*factiva_regions*	gtds_annotations

[1] A note on the use of italics. The tables in italics in the iTRACE Core Tables list are those tables that do not need to be initially populated with data when first put on a new data serving machine. The tables in italics in the iCAST tables are not, strictly speaking, used at the moment. However, they are relatively small and have information we don't wish to lose.

dict_actors	groupdata	factiva_story_codes	gtds_annotations_mapping
dict_sector_frequencies	groups	geonames	intervals
dict_sector_mappings	impactvariable	geonames_specificity_levels	parametertypes
dict_sector_types	impactvariable_statsmap	publisher_mappings	scales
dict_sectors	impactvariable_values	publisher_types	
eventdatafilter	impactvariablestats	sentencesyntaxtree	
events	linearpredictor	serif_events	
eventtypefilter	linearpredictorpointsvalues	*simple_badevents*	
eventtypefilter_ eventtypes	linearpredictorvalues	story_cocom_mappings	
eventtypegroups	lpdpv_points	story_country_mappings	
eventtypegroups_ eventtypes	lpredictor_datapoints	story_native_language_details	
eventtypes	lpredictor_forecast_points		
locations	lpredictor_lags		
publisherfilter	lpredictor_setters		
publisherfilter_ selections	lpredictor_values		
publishers	lpvv_lagstrings		
publishertypefilter	lpvv_lagvalues		
publishertypefilter_ selections	mesaview		
sentences *(W-ICEWS only)*	model_summary_details		
simple_events	model_summary_inputs		
stories	modeldataparameters		
suggestedactors	modeldatapoints		
viewed_events	modelsummary		
	modelsummary_vars		
	moi_results_backup		
	parameterselections		
	predictionpoint		
	predictor		
	runtime		
	studies		
	study_modeldata parameters		
	study_modeldatapoints		
	studycollections		
	timeseriesview		
	timeseries_valueslist		
	variablevalue		
	version		

Table 1 - ICEWS RCDR Tables

At the center of the iTRACE database schema is the concept of Events (representing *who* did *what* to *whom*, *when* and *where*), which are stored in several tables depending on how they are used and what they represent:

- **Events** is the unprocessed table of 'valid' events that will show up in the iTRACE system. It is populated when events are generated in processed, as described in Sections 5.4 and 5.5

- **Badevents** is the unprocessed table of 'invalid' events that will **not** show up in the iTRACE system. It is populated when events are processed with filters, as described in Section 5.5.[3]
- **Serif_events** is a table where events that are generated by Serif are kept. Before they are included in the iTRACE system, they must be migrated from the serif_events table to the events table, as described in Section 5.?.
- **Bad_serif_events** is a table where Serif events flagged as 'historical' and 'ongoing' are moved as part of the Serif event migration task. These events are not included in the iTRACE system and not visible to the user; only Serif events that are flagged as 'current' or 'neutral' are actually migrated to the events table.
- **Simple_events** is the post-processed table of 'valid' events that will show up in the iTRACE system. It pulls in other information about the event as new columns in the table, in order to make querying (and thus, the iTRACE system) faster. It is populated when the simple_events table is created, as described in Section 5.8.
- **Simple_badevents** is an analogous table for the post-processed badevents (not used in Drop 15).

Revolving around these tables are a number of other tables that make up the iTRACE system. Those tables, along with the basic event tables, are explained in detail in the following sections.

3.1 Actor Dictionary Tables

Actors, in the iTRACE sense, are people, groups, or places that participate in events. Actor dictionary tables represent all known information about these actors. This includes their name, synonyms or aliases they are known as, country affiliations, and sector affiliations. Though names and synonyms remain constant over time, country and sector affiliations can and will vary over time, and the schema supports this.

3.1.1 Sectors

A sector can be thought of as a role, in that it describes (optionally time-constrained) properties of an actor. Examples of sectors include government affiliation, ethnicity, religion, the fact that they are a student, the fact that they are associated with a dissident group, etc. Information about sectors are stored in the database tables that begin with '`dict_sector`' in their name, with the addition of the oddly named `dict_agent_sector_mappings` table, and are described below:

Name	Purpose	When Populated
`dict_sector_types`	High level categories of sectors. These are visible in the Event Browser, when the user chooses the 'Get Sources' or 'Get Targets'	When the dictionary is ingested. Typically this table is fairly static. If you wanted to

[3] There exists outside of ISPAN a 'bad event browser', which is similar to the event browser except that it specifically views bad events. This was not deployed to ISPAN. Therefore, the badevents never show up in a system GUI, and are only visible through viewing the raw database tables.

	option. They appear in the 'Properties' tab at the bottom of the Actor Selection Pane.	change how the Event Browser presented its 'Properties', you would modify it.
dict_sectors	The sectors themselves. A subset of these are visible in the Event Browser, in the Actor Selection Pane.	When the dictionary is ingested. Typically this table is fairly static. If you wanted to define new sectors, you would modify this. It will not be of much use unless you modify actors to reference the new sector, though.
dict_sector_frequencies	This is a lookup table, referenced elsewhere to indicate how frequently actors have this sector affiliation. It is used in the Actor Selection Pane of the Event Browser to constrain which sectors are presented to the user (which is necessary as there are over 500 sectors, while less than 100 are commonly used).	It is recommended not to modify this table (though doing so would change which sectors appeared in the Actor Selection Pane of the Event Browser).
dict_sector_actor_ mappings	This identifies cases where an actor represents the entirety of a sector for a given country. For example, it is used to map the fact that for the United Kingdom, the 'Upper House' sector is referred to as 'House of Lords'.	This table is populated as part of a dictionary update task. If an actor is given a sector where the "Use Actor as a proxy of the selected actor" box is checked, an entry noting this fact will be made in this table.
dict_agent_sector_ mappings	This identifies cases where an agent represents the entirety of a sector for a given country. It is similar to the dict_sector_actor_ mappings table, but for	This table is populated as part of a dictionary update task.

	agents.	
`dict_sector_mappings`	This maps every actor to every sector affiliation it has, in a time-dependent fashion.	This table is modified during the dictionary update process.

Table 2 – Sector-related Database Tables

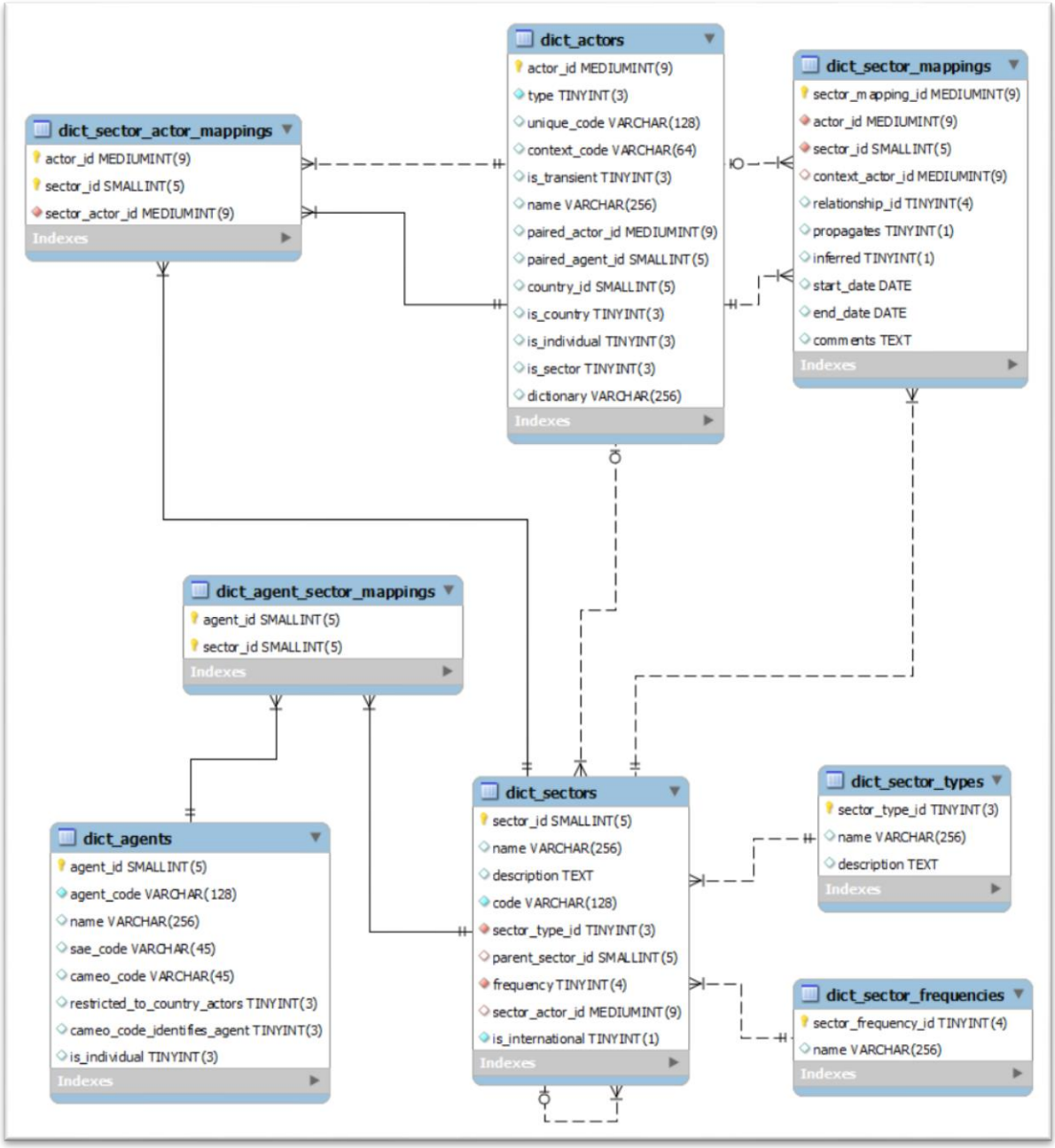

Figure 1 - Sector-related Database Tables

3.1.2 Actors vs Agents

An important distinction in the event coding process is that between actors and agents. An agent is an improper noun, such as 'military', 'dissident', or 'student'. Actors, then, can take one of two forms:

1. Proper nouns such as 'Barack Obama', 'Parliament', or 'Al Qaeda'.
2. Agent-based composite actors, which are comprised by merging a generic agent with a non-composite actor (e.g., an actor of the 'proper noun' type). Examples include 'President (United States)' or 'Dissident (Al Qaeda)'.
3. Sector-based composite actors, which are comprised by merging a sector with a non-composite actor. Examples include 'Government (Honduras)' or 'Religious (Chile)'.

The purpose of having generic agents is one of simplification. Take the concept of a country's military, represented by the generic agent 'military'. One might consider the US military, the Afghanistan military, the Honduran military, etc. Similarly with the generic agent of 'student', where one can have 'UK student' along with 'Italian student', etc. By specifying an agent, you eliminate the need to define 200+ variants of this generic concept in order to make it specific to all the countries in the world.

An interesting note is that, though there are millions of composite agent-based actors possible from pairing agents with 'proper noun' actors, none of them are stored in the database a priori. This is to contrast with proper noun actors, and agents, which are loaded at the time that actor dictionaries are ingested. Since this combinatorial pairing actually represents a relatively sparse matrix, composite actors are created 'on the fly' during the processing of events (Section 5.5).

In general, an agent can be paired with any 'proper noun' actor in the dictionary that represents a group. However, some agents only make sense when paired with a country (e.g., 'Defense Department' would make sense for countries, but not other groups, like news organizations). In these cases, a flag designates that the agent can only be paired with a country, not with a group.

Sector-based composite actors are primarily used for classification purposes when viewing hierarchies of actors. For example, all government-affiliated actors for Paraguay would appear under the 'Government (Paraguay)' sector-based composite actor, which would then serve as a convenient way to specify "all actors that are government-affiliated in Paraguay" in event queries. Typically, sector-based actors do not themselves appear referenced from events. An exception would be when an actor is marked as representing a sector, as described earlier in this section.

Another important concept common to both actors and agents is that of synonyms, known in the actor dictionary as patterns. For example, Barack Obama (a proper noun actor) might also be referred to as 'President Obama', or the Department of Defense (a generic agent) might also be referred to as 'Ministry of National Security'. The iTRACE system uses these patterns when it is coding events.

Some of the important tables that have to do with actors and agents are described below. Looking within the database may give you a better feel as to how these concepts interrelate.

- **dict_actors** stores all of the actors in the system. Initially, during dictionary import, only proper noun actors are ingested. These may be persons, groups (to include countries), or places. If an actor is a country, the `is_country` field is set to 1, and the `country_id` field will reference into the `countries` table. If the actor is an individual, the `is_individual` field is set to 1. If the actor represents a sector, the `is_sector` field is set to 1; see Section 8.1.1 on the `dict_sector_actor_mappings` table for more details. If the actor represents a group, then all of the before-mentioned fields are set to 0. There is no explicit designation of places at present.

  There are several forms of IDs used on the table, which have evolved over time. The primary one used by the iTRACE system is the `actor_id` column, which is referenced directly in the `events` (and other) tables. There is also a `unique_code`, which is primarily used by the data processing system. Finally, there is the `record_gid` which is used by ISPAN (and therefore not present in the W-ICEWS database). Note that the desired state would be a single ID, but this situation has arisen from the organic growth of the system.

  The `type` field indicates what type of actor it is. Type 1 is a proper noun defined explicitly in the actor dictionaries. Type 2 denotes sector-based composite actors, while Type 3 denotes agent-based composite actors.

  The `paired_actor_id` and `paired_agent_id` are used when there are composite actors formed by pairing another actor with an agent. When this occurs, the `is_transient` field is also set to 1.

  The `context_code` field holds the 3-letter ISO code of the country the actor is affiliated. It is set only under certain arcane conditions in the data processing tool, and is best not relied upon. It's a holdover from a legacy system.

  Finally, the `dictionary` field references the name of the dictionary file this actor was imported from.

- **dict_agents** stores information about all the agents in the system. It is fully populated on dictionary import, and changes only during dictionary updates.

  As with the `dict_actors` table, there are three types of unique IDs: the `agent_id`, the `agent_code`, and (for ISPAN only) the `record_gid`. Similar comments apply.

The `sae_code` and `cameo_code` are leftover codes from legacy coding systems, and are best ignored. The same is true of the `cameo_code_identifies_agent` field.

The `is_individual` field is set to 1 when the agent represents an individual instead of a group. When forming agent-based composite actors, the new actor inherits this value.

Finally, the `restricted_to_country_actors` field, when set to 1, indicates that an agent can only be paired with a country-level actor to create a composite actor, not with a group or individual or place.

- **dict_actor_patterns** describes the pattern, or synonyms, for all proper noun actors, in Jabari format. The `actor_id` field pairs it with the appropriate actor. The `pattern_id` is a unique index to make Hibernate happy.

  Patterns should be in all caps, though when interpreted by Jabari they are almost always case insensitive. A pattern is somewhat like a regular expression with a slightly different syntax. First off, patterns are matched starting at the beginning of words only; they must be preceded by whitespace or else the beginning of the text itself. Secondly, there is the concept of 'stemming' in Jabari, whereby a pattern is matched up not only with itself, but with any word that begins with the pattern.

  There are two special patterns in Jabari patterns, the underscore and the equal sign.

  - An underscore within the pattern itself represents whitespace, and defeats the normal stemming rules that Jabari allows when matching patterns. It will still allow matches of plurals and possessives, however. For example, a pattern of 'AMERICAN_CONGRESS' will not only match 'American Congress' as a string, it will also match 'Americans Congress' (plural) or 'American Congressional' due to the stemming allowed. An underscore at the end of a word disallows stemming at the end of the word. As an example of this, the pattern 'AMERICAN_CONGRESS' will match on 'American Congressional districts'; the pattern 'AMERICAN_CONGRESS_' will not, as it disallows the stemmed form of 'Congress' (e.g., 'Congressional') when doing its pattern matching.

    In general, this means that patterns – especially short patterns – most often should end with an underscore. This will prevent errors that can arise from overly aggressive partial pattern-matching. For example, the simple pattern 'US' will match **any** word that starts with the letter 'us' (e.g., 'usual', 'use', etc.), which is highly undesired.

  - An equal sign at the end of a word is used to indicate an acronym. Acronyms are the one exception to the 'case insensitive' pattern matching of patterns, as any pattern ending in an equal sign **must** be matched in all caps. For example,

the pattern 'US_' will still match on the word 'us' (in lower-case) because pattern-matching is generally case insensitive. However, the pattern 'US=' will only match on the letters 'US', in all caps, with no stemming, and no plurals or possessives.

In general, this means that patterns that represent acronyms should end with an equal sign, to ensure that they are only matched when they are found in all caps in the text.

The `country_id` and `unique_code` fields are legacy fields which are unused.

- **dict_agentpatterns** describes the pattern, or synonyms, for all generic agents, in Jabari format. The `agent_id` field pairs it with the appropriate agent. The `pattern_id` is a unique index to make Hibernate happy.

  The `cameo_code` and `sae_code` fields are part of legacy code support, and are unused.

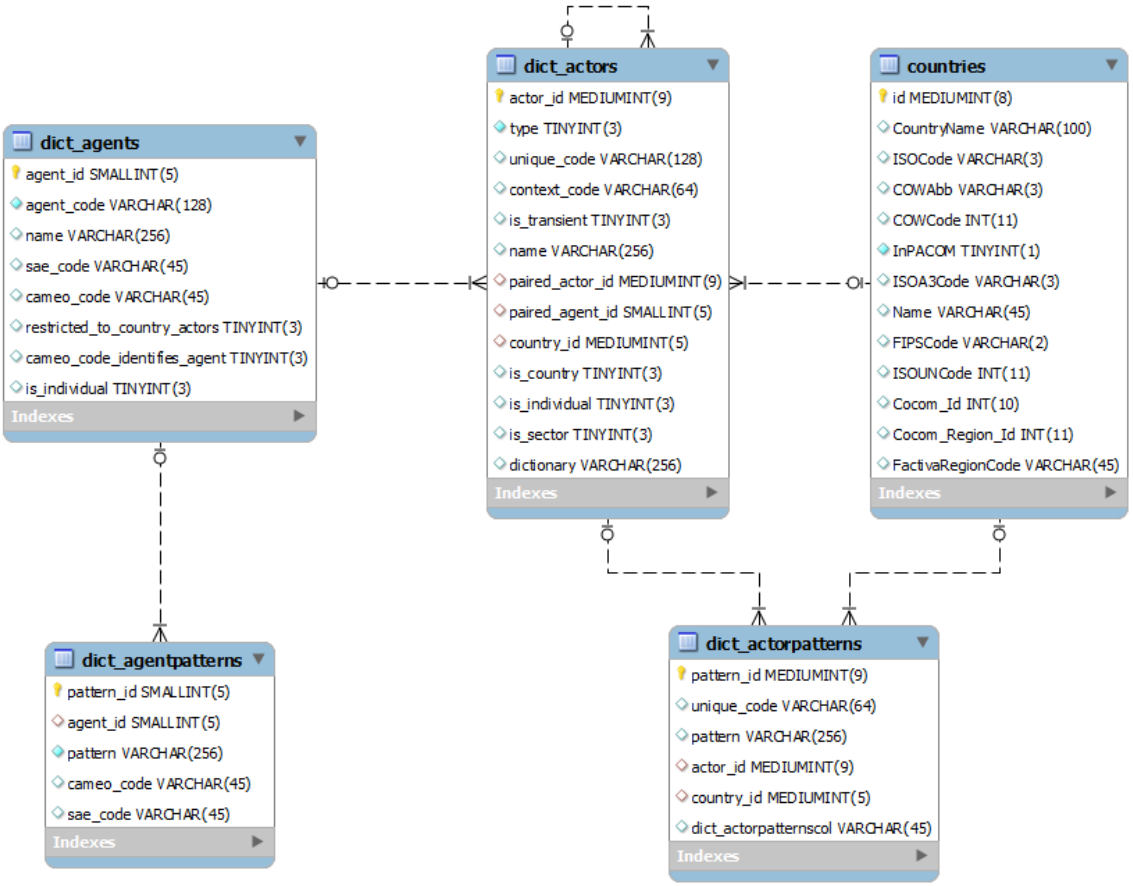

Figure 2 - Actor-related Database Tables

3.1.3 Actor Hierarchies

The concept of a hierarchy of actors is central to iTRACE. In truth there are multiple hierarchies, and these are based around the sector affiliations of actors, which are themselves constrained in time.

Consider an actor like 'Mahmoud Ahmadinejad'. As of 8/3/2005 he is the President of Iran, as can be represented in one time-dependent actor hierarchy that leads from Ahmadinejad, up to Executive Office (Iran), up to Executive (Iran), up to Government (Iran), and ultimately up to Iran. From 6/20/2003 to 8/2/2005 he was the Mayor of Tehran, and this has its own hierarchy that leads ultimately up to Iran. From 1/1/1997 to 6/19/2003 he was a professor at an Iranian university, which is a third hierarchy leading up to Iran. And since 4/11/2003 he has been the leader of the Alliance of Builders of Islamic Iran, which is a fourth hierarchy.

The database represents this hierarchical information in two different tables – `dict_actorlinks` and `dict_actornodes` – which represent the same information but in different formats, as described below.

- **dict_actorlinks** represents the hierarchy as timestamped parent-child links. The format simply has a `child_id` and a `parent_id`, both of which reference the `dict_actors` table, and an optional `min_date` and `max_date` for time constraining the relationship. The `link_id` field simply exists to make Hibernate happy.

- **dict_actornodes** represents the hierarchy in a nested set[4] format, which allows for hierarchical queries without the need for recursion. It not only keeps track of the `parentnode_id` but also the `rootnode_id` and, where known, the `country_id` of said root node. As with the actor links, there is an optional `min_date` and `max_date` for time constraining the relationship. The `nsLeft` and `nsRight` fields are required in the nested set algorithms for representing the hierarchy. The `node_id` field simply exists to make Hibernate happy.

[4] http://en.wikipedia.org/wiki/Nested_set_model

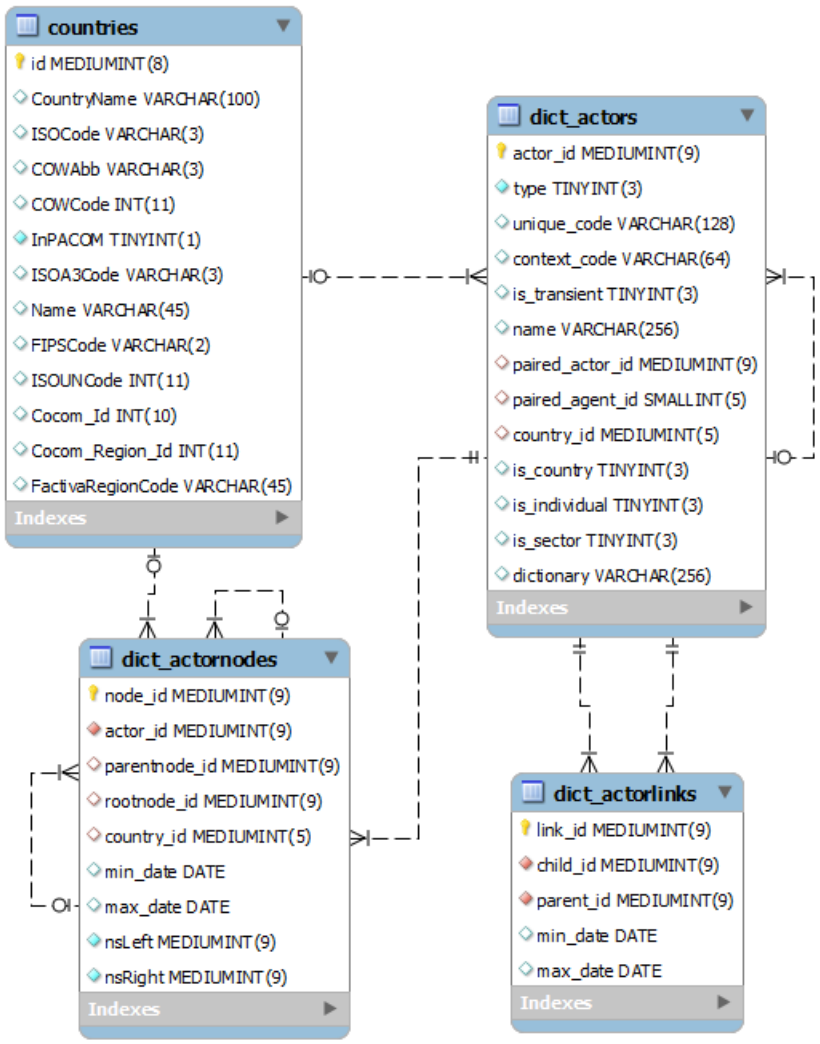

Figure 3 - Actor Hierarchy-related Database Tables

Due to the way the nested set algorithm works, adding and deleting items in the hierarchy can affect the nsLeft and nsRight fields on potentially many rows of data. Whenever a new actor is added, at least one node is affiliated with it, so adding actors in essence affects this hierarchy. Though actor deletions do not occur outside of dictionary updates, new actors are added frequently in the form of composite actors formed by associating a generic agent with a proper name actor. Instead of constantly updating the values in the hierarchy with each addition, the hierarchy itself is recomputed after a batch of actors is added, as happens in the weekly interim event coding. In general, generating the events adds new actors and therefore new values to the `dict_actorlinks` table, but `dict_actornodes` are only added when the entire table is regenerated. This corresponds to Section 5.6 of the document.

Useful things to understand about the actor hierarchy include:

- There are actual multiple hierarchies for any given actor, due to different roles that actor plays over time. Each actor has at least one hierarchy, and therefore, at

least one dict_actornodes and one dict_actorlinks row associated with it.

- The dict_actorlinks and dict_actornodes tables represent the same information in different formats, geared toward different types of data access.
- The dict_actorlinks table is actively maintained as actors are added. The dict_actornodes table is generated all at once, after a batch of actors are added. It is therefore possible for the dict_actornodes and dict_actorlinks tables to be out of sync at a snapshot in time, in which case the dict_actorlinks table would hold the correct data. The number of dict_actorlinks associated with an actor is therefore the same as the number of dict_actornodes associated with an actor.
- These multiple hierarchies can be optionally constrained by time.
- The node_id of dict_actornodes do not persist when the dict_actornodes table is regenerated, and will differ each and every time. This is a direct contrast to the dict_actors table, whose actor_id field values do persist over time.
- The dict_actorlinks for an actor correspond to the sector memberships as defined in the Dictionary Update, for non-composite actors.

3.2 Geographic Tables

Geographic tables are tied to the concepts of COCOMs, COCOM regions, countries, provinces, districts, and cities.

3.2.1 Countries, COCOMS, and Regions

There are several tables that define information about countries, COCOMs, and COCOM regions. They are defined in Table 3 below.

A source of potential confusion is that there are two ways to map countries to COCOMs and their regions, which are in use in varying parts of the system.

- The most direct way this is referenced is in the countries table, via the fields for cocom_id and cocom_region_id, which map into the cocoms and cocom_regions tables directly. In this manner, each country maps to one (and only one) COCOM and region.

- A second way is through the cocom_actor_mappings table. This table is based, in part, off the countries table (by matching the name field of the countries table with the name field of the dict_actors table). It is then augmented with several entries. This work is done in the Stored Procedure within the database that is called COCOM_ACTOR_INSERT. It is primarily used to: a) map regions (e.g., Kashmir, Southeast Asia) to COCOMs; or, b) match certain countries to multiple COCOMs (e.g., ‘Russian Federation’ to both USPACOM and USEUCOM, who share joint responsibility).

Name	Purpose	When Populated
countries	Defines all countries, current and historical. Includes information about codes and COCOM affiliations.	Once at the beginning of database creation. Afterward, adjusted manually.
countrynamechange	Defines aliases for countries that made be used in 3rd-party data sources	By hand.
cocoms	Defines all geographical COCOMs, plus an International division	Once at the beginning of database creation.
cocom_regions	Defines all regions within a COCOM, for the purpose of divvying up the countries at a finer grain level.	Once at the beginning of database creation.
cocom_actor_ mappings	Maps the dict_actors records that indicate countries to their appropriate COCOM and COCOM region	With each weekly event coding.

Table 3 – Countries-, COCOMs-, and Region-related Database Tables

The use of COCOMS is obviously tied to a military view of the world. If one wanted to modify ICEWS into a civilian one, it would be possible to use the cocoms table to represent continents, and the cocom_regions table to represent regions within a continent. Really what the tables represent, names aside, is a way of grouping countries into categories and sub-categories.

The countrynamechange table is used to define alternate ways in which a country may be referenced in 3rd-party data sources. For example, there exists in the countries table an entry with a **name** and **countryname** field both defined as 'North Korea'. If one were to look in the countrynamechange table, they would find alternate ways in which the country be referred: North_Korea, Korea North, N. Korea, People's Republic of Korea, etc.

Though the cocoms and cocom_regions tables are fairly straightforward with what the fields represent, the countries table has a lot of fields with information used at various stages in the ICEWS process. These include:

- **id** is the unique ID used within iTRACE
- **Name** is the name that is shown in the user interfaces. Change this – along with the corresponding dict_actors name field – if you want to change the name of a country.
- **ISOCode** is the ISO 2-letter country code
- **COWAbb** is the 3-letter Correlates Of War (COW) abbreviation
- **COWCode** is the numeric COW code

- **ISOA3Code** is the ISO 3-letter country code
- **FIPSCode** is the Federal Information Processing Standard (FIPS) code
- **ISOUNCode** is the ISO UN code
- **FactivaRegionCode** is the code that Factiva uses to reference the country
- **cocom_id** maps to the singular COCOM this country resides in
- **cocom_region_id** maps to the singular COCOM region this country resides in
- **CountryName** is used in some data processing tasks, and represents a variant on the **Name** field

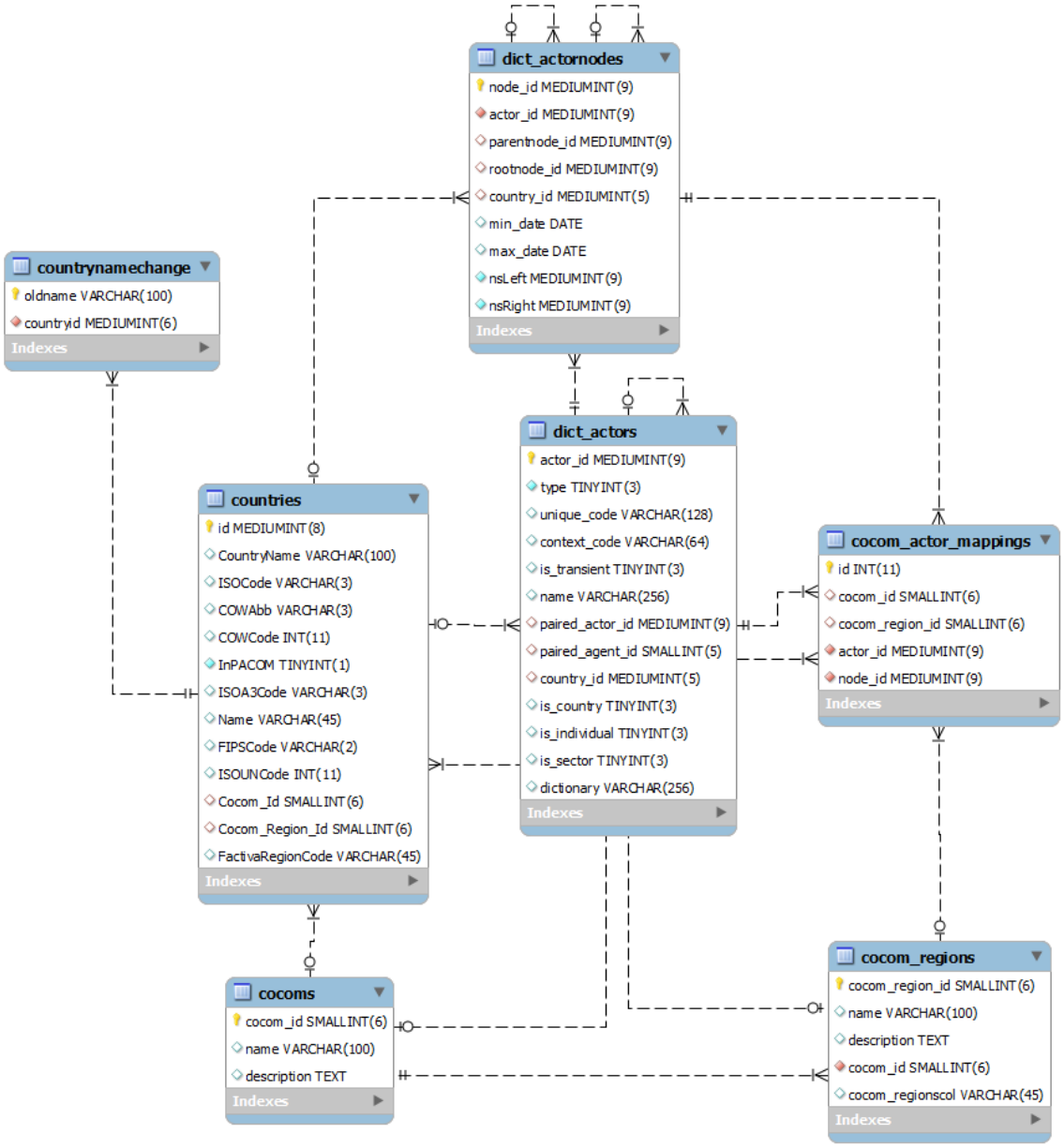

Figure 4 – Countries-, COCOMs-, and Region-related Database Tables

3.2.2 Gazetteer Tables

Gazetteer tables are used in the geolocation task described in Section 5.9 of this document. They are primarily adapted from the geonames gazetteer[5]. These tables are defined in the table below.

Name	Purpose	When Populated
`country_info`	Holds information related to countries	Once at database creation
`geonames`	Holds geographic information below the country level	Initially at database creation. Can be manually extended.
`geonames_specificity_levels`	Reference via the foreign key `specificity` in the `geonames` table	Initially at database creation.
`altnames`	Holds information about synonyms for geographic information below the country level	Initially at database creation. Can be manually extended.
`locations`	Holds information about locations pertaining to events	Initially seeded with country-level locations. Augmented dynamically during the Geolocation task. This is **not** a geonames table, and is ICEWS specific.

Table 4 – Gazetteer-related Database Tables

Details about the geonames tables can be found on the Internet. As of 12/6/2011 the URL for the descriptions is: http://download.geonames.org/export/dump/

The `locations` table has fields as follows:

- `location_id` is the unique ID used by iTRACE.
- `entityText` is the text string that is used to identify this location.
- `city` is filled in if a city is associated with this location. A district, province, or country-level location will have this value as NULL.
- `district` is filled in if a district is associated with this location. A province or country-level location will have this value as NULL. Sometimes, even a city will have this value as NULL, when the gazetteer could not match up a district for the city.
- `province` is filled in if a province is associated with this location. A country-level location will have this value as NULL. Sometimes, even a city or district

[5] http://www.geonames.org

will have this value as NULL, when the gazetteer could not match up a province for the it.

- `area` refers to a geographic area of a country, such as 'southwest Afghanistan' or 'northern Iran'. It is filled in if a geographic region below the country-level, but above the province level, is detected.
- `country_id` maps to the `id` field of the countries table.
- `latitude` and `longitude` are exactly that. Sometimes these values are not known in the gazetteer. For countries, the latitude and longitude used is that of the capital city of the country.
- `geonameid` maps to the `geonameid` field of the `geonames` table. This is not present for country-level or area-level locations, but should be present for all others.

The `geonames` and `altnames` tables are taken as a subset of geonames and only slightly modified.[6] There are a few added columns to the `geonames` table, however, as follows:

- `is_common` is set to 1 for cities which, subjectively speaking, are common enough to be used as a point of reference. It is used by the geolocation algorithm.
- `is_capital` is set to 1 for capitals of countries, and 0 otherwise.
- `is_topcity` is set to 1 for the five most populous cities in a country, and set to 0 otherwise.
- `specificity` is set to 0 for cities, -1 for geographic locations within a city (squares, mosques, hospitals, etc.), 1 for districts, 2 for provinces, and 3 for areas of a country (northern Afghanistan, southeast Afghanistan, etc.).
- `lower_name` is set to whatever the all-lowercase version of the `name` is.
- `lower_asciiname` is set to whatever the all-lowercase version of the `asciiname` is.

[6] There is a related document on geolocation that describes the process of deciding what from geonames to include in the ICEWS system.

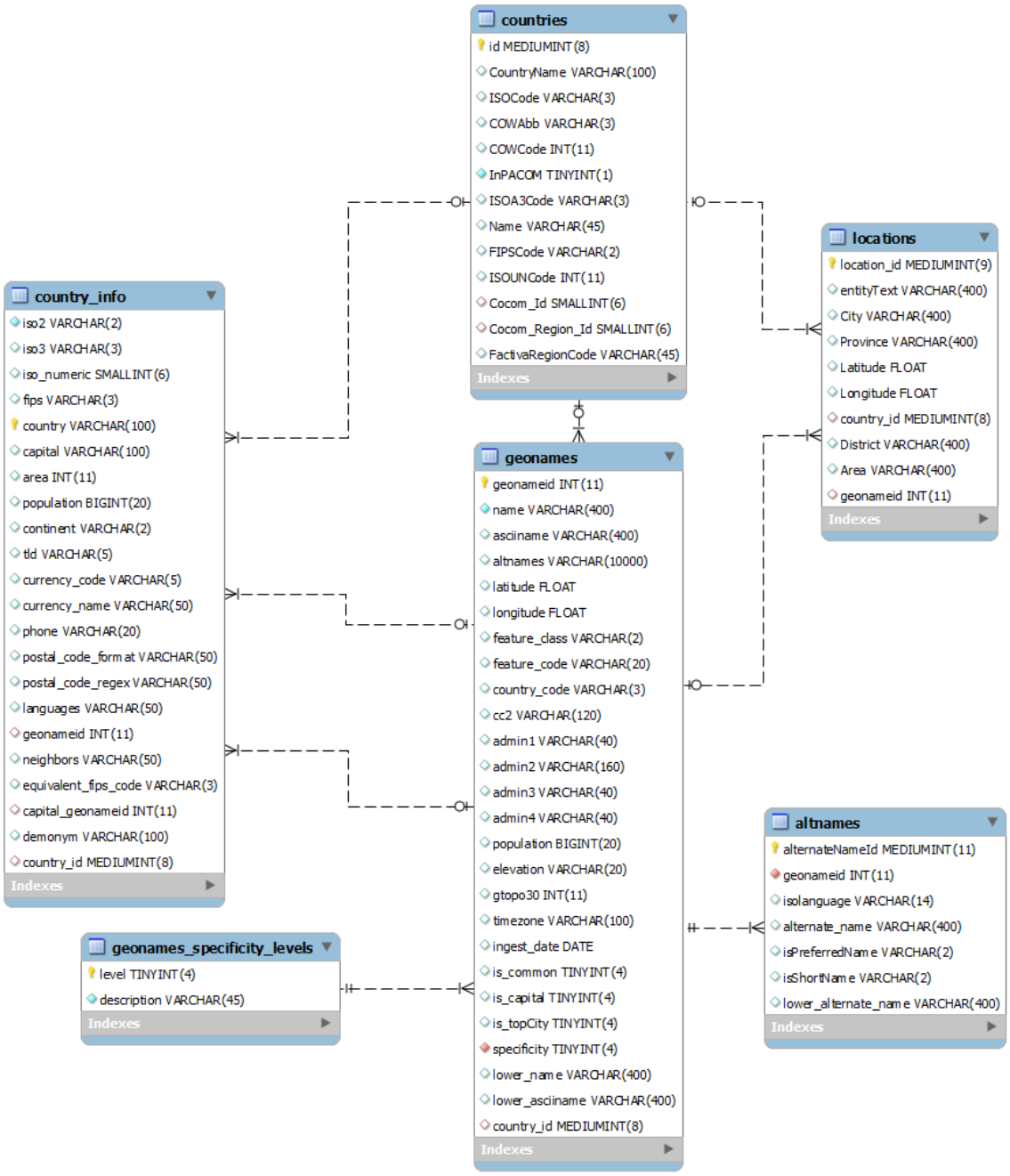

Figure 5 – Gazeteer-related Database Tables

The `altnames` table has two fields, `isshortname` and `ispreferredname`, which are legacy fields and no longer used. The `lower_alternate_name` field is simply an all-lowercase version of the `alternate_name` field.

3.3 Story Related Tables

These tables have to do with information specifically related to stories, which refer to news stories specifically.

3.3.1 Publisher Tables

The tables related to publisher information are the `publishers` table, the `publisher_types` table, and the `publisher_mappings` table. These tables are maintained by hand.

The `publisher_types` table is fairly straight-forward. It's used to classify publishers into different types, and is shown in the user interface. There is probably hard code in the ICEWS system that relies on this table, so changing it is not advised without a thorough knowledge of the system and what effects it might have. Currently, ICEWS has two publisher types: one for stories obtained from Factiva, and one for stories obtained from the OSC.

The `publishers` table is also fairly self-explanatory. You can change the `name` or `description` field for a particular publisher if desired, and change its country affiliation as well with the `country_id` field. The `pub_type_id` field references the `publisher_types` table described above. Finally, the fairly cryptic `is_group` field is used to denote (by setting the bit to 1) the publisher which should be used if a story references a publisher who is not found in the `publishers` table. There should ideally be only one value to set to 1 for each publisher type.

The `publisher_mappings` table maps synonyms for publishers (represented in the `publisher_text` field) to the canonical form of the publisher (referenced in the `publisher_id` field, which maps to the `id` field of the `publishers` table). The `publisher_name` field is not really needed as it is extraneous information. The `publisher_mapping_id` is simply there to make Hibernate happy. Every publisher should have at least one `publisher_mapping` table entry, where the `publisher_name` and `publisher_text` fields are identical. Additional entries would represent aliases or synonyms for which the publisher is also known.

3.3.2 Other Story Related Tables

There are a number of stories related to the stories themselves, as opposed to the events. They are summarized in the table below.

Name	Purpose	When Populated
`stories`	Primary table holding stories (both English and foreign language)	Ingestion of New Stories and Conversion to XML
`badstories`	Holds suspected duplicate stories (same publisher, same date, same headline)	Ingestion of New Stories
`story_native_language_details`	Supplementary table that holds info on foreign language stories	Ingestion of New Stories
`factiva_story_codes`	Metadata information about ingested stories	Ingestion of New Stories

factiva_regions	Metadata specific to Factiva's regional tagging of stories	Initially on database creation.
factiva_region_ hierarchy	Metadata specific to Factiva's regional tagging of stories	Initially on database creation.
sentencesyntaxtree	Holds the syntax trees used in event coding and geolocation	Event generation
story_cocom_ mappings	Maps stories to one or more cocoms	Ingestion of New Stories
story_country_ mappings	Maps stories to one or more countries	Ingestion of New Stories
sentences	Contains a reference to every sentence from which an event was coded	Sentence generation

Table 5 – Story-related Tables

The story_cocom_mappings and story_country_mappings are fairly straightforward. They are set by looking at information provided in the raw stories about what countries a story pertains to.

The factiva_story_codes table simply records the story codes associated with a given story; despite the name, it also holds metadata information for OSC stories. There is no known taxonomy to describe what the codes mean, and they are not currently used in the system, simply stored in case they are useful in the future. However, certain regional information related to Factiva stories only are stored in the factiva_regions and factiva_region_hierarchy tables.

The stories table is set in stages, and has a number of fields, described in the table below. The badstories table is very similar, and is where a story is removed when it is likely to be a duplicate of another story from the same publisher. Within the badstories table, the original_story_id field is a foreign key into the stories table.

Field	Description	When Set
Storyid	Unique field used by iTRACE	Ingestion of New Stories
Publisher	Publisher according to Factiva/OSC. Often this is not set. This is actually different from what ICEWS refers to as a story's publisher, and is not directly used by the system.	Ingestion of New Stories
PublicationDate	When the story was published	Ingestion of New Stories
IngestDate	When the story was imported into the database. It is for reference only.	Ingestion of New Stories
Headline	The headline of the story. For foreign language stories this will be in the foreign language. Occasionally, if the headline is not	Ingestion of New Stories

	properly described in the story XML, this value will be null	
RawText	The raw text of the story, in English. There are several circumstances when this might be null: a) if a story is in a foreign language and has not yet translated; or, b) if a raw story's XML simply could not be parsed. This is used in setting up the indexes used in the Story Search and Concept Tracker portlets.	Ingestion of New Stories
XMLText	The text in XML format. This field will always be null in the case where the RawText is null. It might also be null if the OpenNLP package could not delineate the sentences.	Conversion to XML
Status	Coding status of the story. For reference Only	Both
Source	The Source string is specified by a human being at the time the story is imported. It is used to keep track of which batch the story was part of.	Ingestion of New Stories
Header	The header, as specified by Factiva/OSC. This is infrequently set, and is not currently used in the system.	Ingestion of New Stories
Comments	Log of actions performed on stories. It is for reference only.	Both
Filter	Deprecated and no longer used.	N/A
Original_story_id	Deprecated and no longer used	N/A
Canonical_ publisher_id	References into the publisher_id field of the publishers table. It is set by cross-referencing the SourceName field with the publisher_text field on the publisher_mappings table.	Ingestion of New Stories
FactivaAccessionNo	Unique ID used by Factiva (and OSC, despite its name). We use this to detect duplicate stories.	Ingestion of New Stories
SourceName	The source name, as set by Factiva/OSC. This is actually what we use for our publisher.	Ingestion of New Stories
SourceCode	Source code, as set by Factiva/OSC. It is frequently null and is currently unused.	Ingestion of New Stories
Byline	Byline, as set by Factiva/OSC. It is frequently null, and currently unused in the system.	Ingestion of New Stories

Table 6 - Stories Table Fields

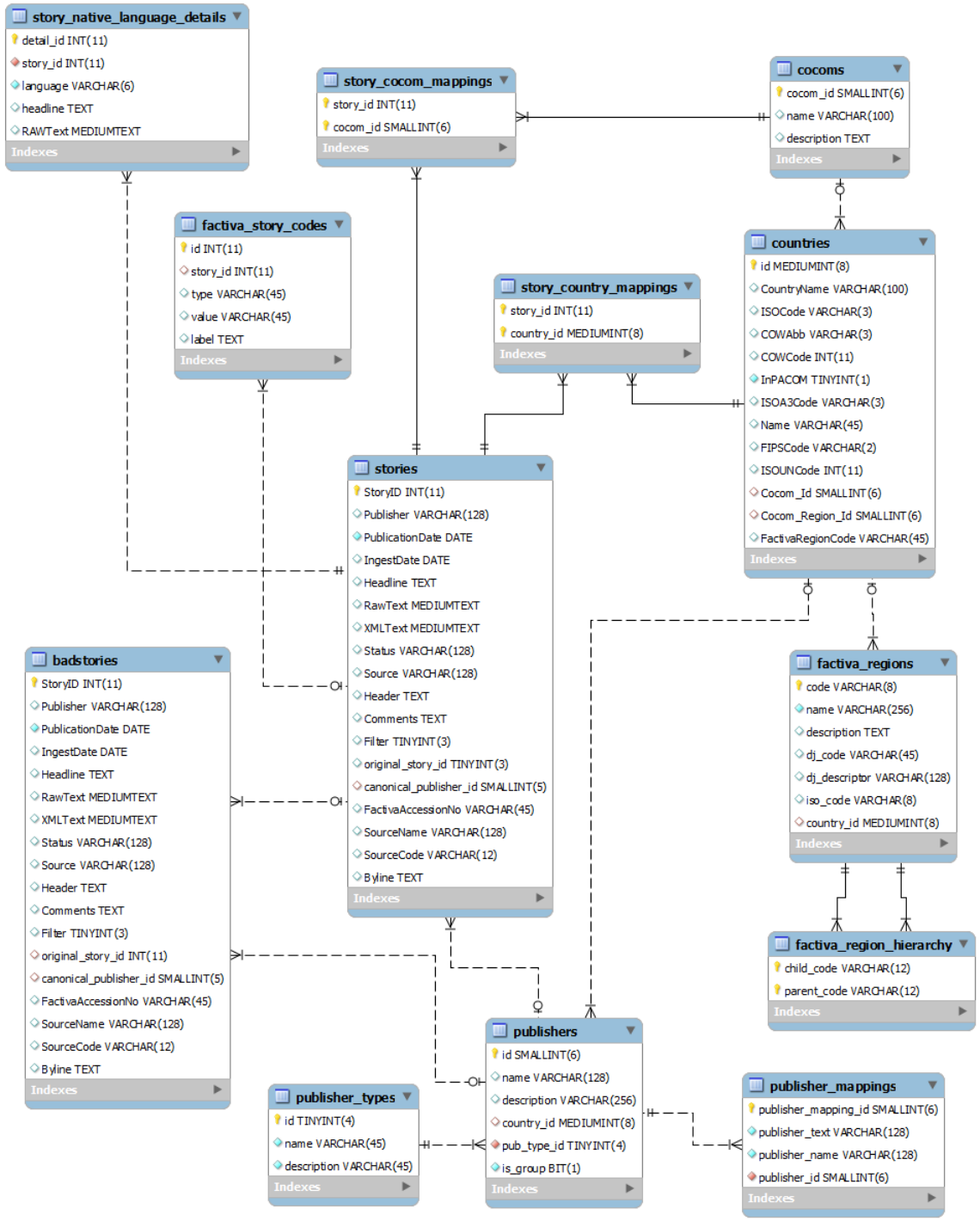

Figure 6 - Story-related Database Tables

The `story_native_language_details` table holds native language information about foreign language stories. Where the stories table only holds English text, this table holds only foreign language text. Most stories that have a null value for their `RawText` field will have a corresponding entry in this table, with the exception being stories that were not parsed properly from their XML. Currently the only values for the `language` field are ‘es’ (for Spanish) and ‘pt’ (for Portuguese).

The `sentencsyntaxtree` table holds syntax trees for every sentence that has been previously event coded. This information is cached because generating a syntax tree is a time-consuming process that only has to be done once per story ingestion, not once per event coding. That means, if the events are ever recoded, the syntax tree generation does not need to be recomputed, saving time.

The syntax trees are generated by the OpenNLP package and use a syntax similar to that of the Penn Treebank parser[7]. Sometimes a sentence is too difficult to parse, and there will be no reference to it in this table. In that case, during a recoding the system will attempt to parse it again (and fail again). This generally does not happen often, so it has a negligible impact on performance.

The `sentences` table is only used in the W-ICEWS system, and is not found in the G-ICEWS/ISPAN version of the software. It holds a copy of every sentence from which an event is coded.

3.4 Event Related Tables

Event related tables refer to the tables to do with coded events that aren't classified elsewhere.

3.4.1 The Eventcoders Table

The `eventcoders` table has very basic information about the different event coders used by the ICEWS system; currently these coders are JabariNLP and Serif. It is referenced via a foreign key from various event-related tables.

3.4.2 The Eventtypes Table

The eventtypes table is directly tied into Jabari's CAMEO verb dictionary, and as such, it is advised not to modify this table. Information is taken from the CAMEO Codebook[8], and modifying it will deviate from that publication. Fields are described as follows:

- **eventtype_id** is the primary unique ID used by iTRACE.
- **name** is what the event type is name. These are taken from the CAMEO codebook directly. You may rename the event types here, but be aware that you will be deviating from the CAMEO standard. You should also maintain the meaning of the event type, should you rename it. You would also want to update the Event Browser's, and iTRACE's, help text.
- **code** represents the CAMEO event type code for this event type. It is referenced within the verb dictionary, so you should under no circumstances change these values unless you are making extensive changes to the verb dictionary, the scope of which is beyond this document. You would also want to update the Event Browser's, and iTRACE's, help text.

[7] http://www.cis.upenn.edu/~treebank/
[8] http://web.ku.edu/~keds/cameo.dir/CAMEO.CDB.09b5.pdf

- **goldstein** is the Goldstein value of the event type, also known as the event type's intensity. It ranges from -10 to 10, where negative numbers are hostile actions and positive numbers are cooperative actions. The more extreme the value, the more hostile (or cooperative) the event type is. Changing these numbers will deviate them from the CAMEO standard, and is not advised. Should you do so, you would also want to update the Event Browser's, and iTRACE's, help text.
- **nsLeft** and **nsRight** are fields that are used to represent the hierarchy of event types, using the nested set format described in Section 8.1.3 for the `dict_actornodes` table. It is strongly advised not to modify these values, as you would be changing the hierarchy in probably unintended fashions.
- **description** is the description of the event type. It is taken from the CAMEO codebook directly, and appears as a tooltip within the Event Browser's Event Type Selection Pane.
- **usage_notes** are the usage notes for event types. They are taken from the CAMEO codebook directly, and appears as a tooltip within the Event Browser's Event Type Selection Pane.
- **example** are the examples for event types. They are taken from the CAMEO codebook directly, and appears as a tooltip within the Event Browser's Event Type Selection Pane.

3.4.3 Events and Badevents Tables

These two tables are central tables in the iTRACE system, and contain the event data used throughout the iTRACE system. They reference previously described tables extensively. Information is appended to these tables in the Generate Events and Processing Events steps of event coding.

The events table holds events that are actively used by the iTRACE system. The badevents table holds events which have been filtered out in the processing of events, due to one of the filters. Refer to Section 5.5 on processing events for details on these filters.

Since the fields are very similar across the two tables, they will be described in the table below. In the event that a field only applies to one of the two tables, it will be noted.

Field	Description	When Set
`event_id`	Unique ID used by iTRACE	Event generation
`source_actor_id`	Source actor; reference `actor_id` field of `dict_actors` table	Event processing
`source_actor_pattern_id`	Source actor pattern; references `pattern_id` field of `dict_actorpatterns` table	Event generation
`source_agent_pattern_id`	Source agent pattern, if a composite actor; references `pattern_id` field of `dict_agentpatterns` table	Event generation

target_actor_id	Target actor; reference actor_id field of dict_actors table	Event processing
target_actor_ pattern_id	(events table only) Target actor pattern; references pattern_id field of dict_actorpatterns table	Event generation
target_agent_ pattern_id	Target agent pattern, if a composite actor; references pattern_id field of dict_agentpatterns table	Event generation
eventtype_id	References eventtype_id field of eventtypes table	Event generation
verbrule_id	(events table only) Deprecated and not set	N/A
verbrule	(events table only) Shows the text of the verb rule from the verbs dictionary that generated this event	Event generation
story_id	References the id field of the stories table	Event generation
sentence_num(ber)	References the sentence number in the XML Text that the event was generated from	Event generation
text	(events table only) For Jabari-generated events, the text in the story that was identified as generating this event; for Serif-generated events, the verb tense	Event generation
event_date	The date that the event occurred on; for simplicity, it is taken to be the publication date of the associated story	Event generation
location_id	The location where the event occurred, if known	Geolocation
coder_id	Set to 1 for Jabari-generated events, and 2 for Serif-generated events	Event generation
flag_event_date	Used by the 'flag bad event' functionality that is not being deployed in Drop 15	N/A
comments	(badevents table only) Set when an event is moved from the events to the badevents table	Event processing

Table 7 - Events and Badevents Tables Fields

Which many fields are set in event generation, a few fields are set in the event processing stage. Notable amongst these are setting the source_actor_id and the

`target_actor_id`. The reason this is done separately is that composite actors, when first encountered, are created dynamically. In a single-threaded system, creating actors on the fly is not a problem. However, when multiple event generation threads are run at the same time – as is typical when the entire corpus of stories is being recoded – creating new actors within multiple threads could result in race conditions. In particular, some composite actors might be inadvertently duplicated.

To work around this, the process of coding events was divided into two parts: the generation part (which can be run in parallel) and the processing part (which cannot, without certain caveats). The generating part only records information about the *patterns* that were found in events. In this manner, no composite actors are ever created. It is the processing step that goes through sequentially and where necessary, in a single thread, creates new composite actors. Filtering is done in the processing step and not the generation step not because it is parallelizable, but because the filtering process needs knowledge of the actual actor IDs to work.

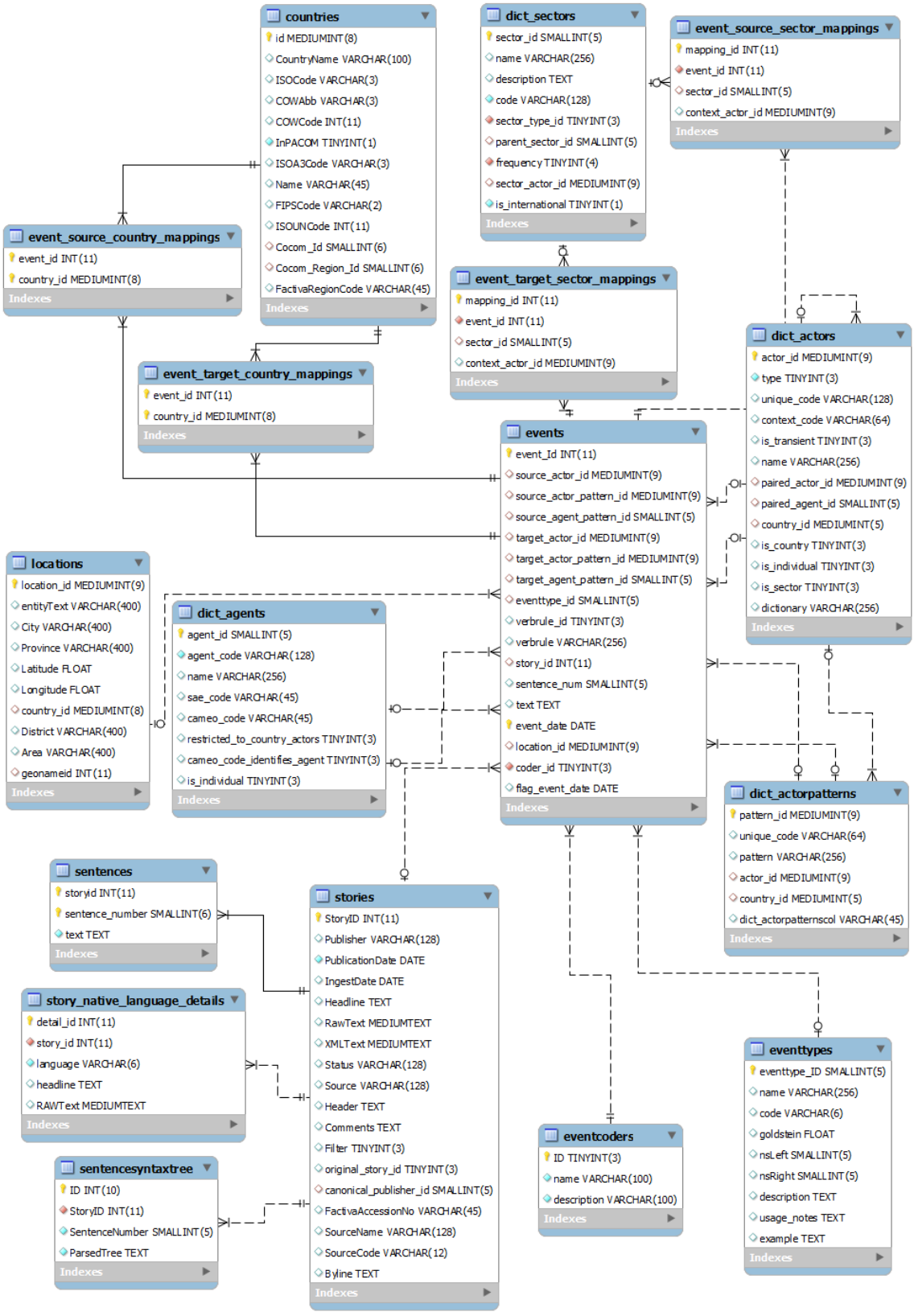

Figure 7 - Event-related Database Tables

3.4.4 Simple_events and Simple_badevents Tables

These two tables are the results of post-processing the `events` and the `badevents` tables, respectively. They augment the fields of these tables with additional information, while omitting other fields; when set in advance these modifications can make iTRACE queries run much faster as opposed to running off the `events` or `badevents` table and performing the relevant JOIN statements during runtime.

As most of the values on these tables are set via simple JOIN statements, the best way to understand the fields in these tables is to look at the stored procedures that generate them. These are `SIMPLEEVENTS` and `SIMPLEBADEVENTS`, respectively.

The only thing that is not straightforward is how the relevant `dict_actornodes` for the source and target actors are selected. As indicated in Section 8.1.3 on actor hierarchies, the `dict_actornodes` table captures actor hierarchy information in a format that is easy to query efficiently. The complication arises from the fact that a particular actor may have multiple actor hierarchies, constrained over time, that they are involved in. The JOIN needs to pick the most appropriate one.

This is done with the database function `RETURNNODE`. In general, the only nodes considered are either those with unconstrained dates (i.e., their `min_date` and `max_date` fields are null) or ones where the event date falls within the `min_date` and `max_date` boundaries. This may still lead to multiple nodes to consider, as an actor can reside in multiple hierarchies at any point in time. Precedence is given to the most restrictive date boundaries, where possible, for the sake of simplicity.

Field	Description	When Set
`source_nsLeft, source_nsRight`	Used for a nested set hierarchical description of source actor	Simple Event Generation
`source_country_id`	ID of source actor's country affiliation	Simple Event Generation
`source_parent_actor_id`	ID of source actor's parent actor	Simple Event Generation
`source_root_actor_id`	ID of source actor's root actor up the hierarchy; typically a country-level actor's ID	Simple Event Generation
`target_nsLeft, target_nsRight`	Used for a nested set hierarchical description of target actor	Simple Event Generation
`target_country_id`	ID of target actor's country affiliation	Simple Event Generation
`target_parent_actor_id`	ID of target actor's parent actor	Simple Event Generation
`target_root_actor_id`	ID of target actor's root actor up the hierarchy; typically a country-level actor's ID	Simple Event Generation
`canonical_publisher_id`	The ID of the publisher of the story	Simple event

	from which the event came	generation
pub_type_id	The ID of the type of the publisher of the story from which the event came	Simple event generation

Table 8 - Augmented simple_events and simple_badevents Tables Fields

3.4.5 Serif_events and Serif_badevents Tables

The serif_events table is where the Serif event coder stores its events, which must then be migrated by the system into the standard events table. Must of the fields are similar to what is found in the events table, though there are extra source- and target-related fields as not all of the actor resolution is done within Serif. The event_tense field stores information specific to the Serif coder and is used to indicate if events are considered historical, ongoing, current, or neutral. The event_id of this table is a Serif-specific event ID, while the icews_event_id is a foreign key into the events table, and set at the time that the serif_event is migrated.

Also during Serif event migration, Serif events with an event_tense of 'historical' or 'neutral' are not migrated to the events table, but instead moved to the bad_serif_events table. This is so that only specific current events are ingested into the ICEWS system.

3.4.6 Event Mapping Tables

The event mapping tables consist of the following four tables:

- event_source_country_mappings: maps events to a country (usually one, sometimes multiple or none) based on the affiliation of the source actor of the event
- event_target_country_mappings: similar to the above, but the determination is based on the affiliation of the target actor
- event_source_sector_mappings: maps events to typically multiple sectors based on the affiliations of the source actor
- event_target_sector_mappings: similar to the above, but based on the target actor affiliations

These tables are only used during data processing, and are set during the processing events. Note that if the source or target is an international actor for a particular event, there will not be an entry in that respective table for the given event_id, as the value would simply be null.

Additionally, the tables that map to sectors explicitly represent sector hierarchies as multiple entries. For example, consider a source actor who is affiliated with the Executive Office sector. This sector has a parent sector of the Executive sector, which in turn has its parent as the Government sector. In the event_source_sector_mappings table,

then, there would be three entries: one mapping the event to the Executive Office sector, one mapping it to the Executive sector, and one to the Government sector.[9]

3.5 Event Aggregation Tables

A number of tables are used to support aggregating event data in ICEWS, which is done as a pre-processing step in the data processing cycle. For a complete understanding of ICEWS event aggregations, refer to the Event Aggregations on ICEWS document.

Aggregations are defined in the Actor Dictionary tool, and then imported into the database through use of the ICEWS Data Tool. The following tables are involved:

Name	Purpose	When Populated
`dict_eventqueries`	Every event query has an entry in this table, which describes metadata about the query.	ICEWS Data Tool initqueries task
`dict_eventqueryconstaints`	This table defines sector- and country-related constraints that are used to define how to filter events for an event query.	ICEWS Data Tool initqueries task
`dict_seccons_sec_mappings`	When a query constraint involves one or more sectors, they are referenced in this table.	ICEWS Data Tool initqueries task
`dict_eventquery_methods`	This table describes how filtered sets of events should be post-filtered and aggregated to come up with a numeric total.	ICEWS Data Tool initqueries task
`dict_eventquery_method_types`	This table contains the four numerical methods in which event intensities can be aggregated.	On database initialization.
`dataparam_events_mapping`	For every event query of interest, the set of filtered and post-filtered events comprising the aggregation are stored.	ICEWS Data Tool event aggregation task

Table 9 - Aggregation-related Database Tables

[9] The primary purpose of the sector mapping tables is to allow for queries to get the full set of sectors that are involved in an event, without having to devise some sort of recursive query.

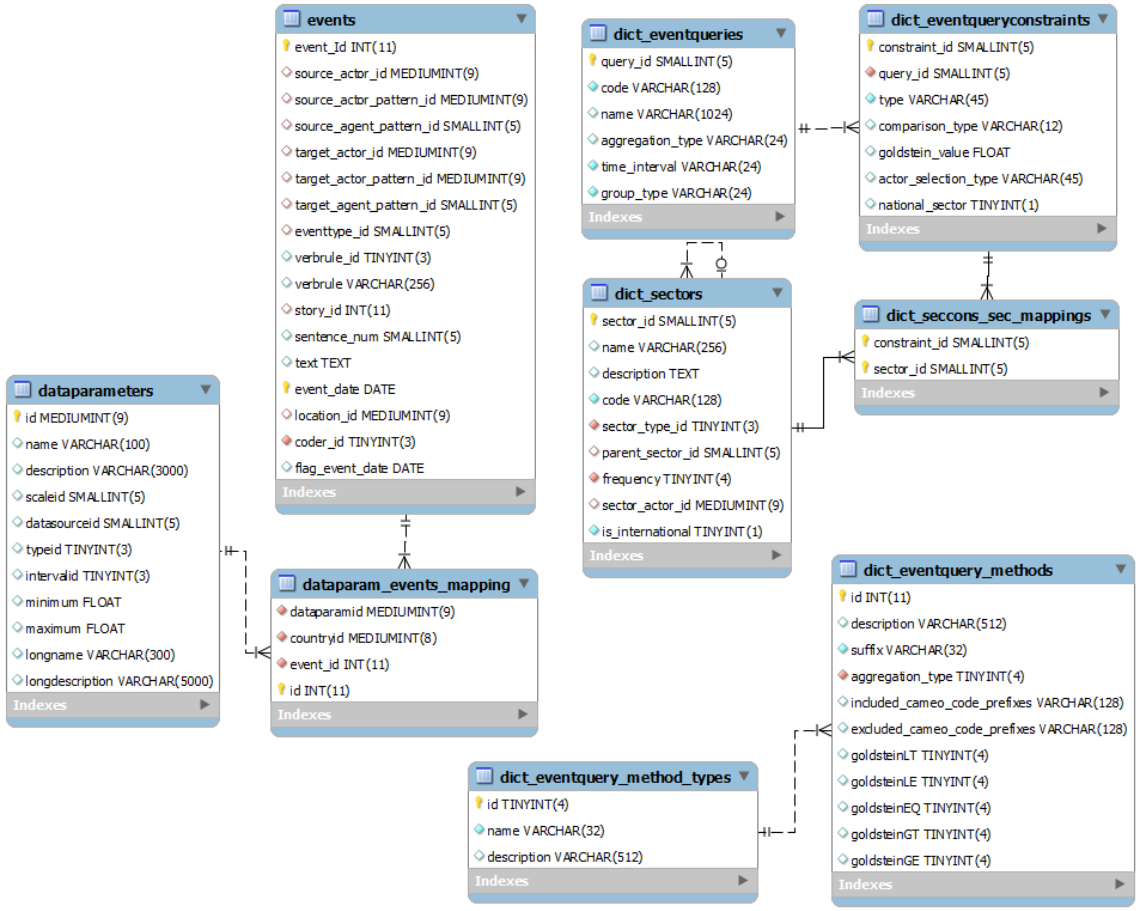

Figure 8 - Aggregation-related Database Tables

3.6 State Data Tables

State data is most often numerical data that is associated with a particular country (or in geopolitical terms, a particular state) for a given *time interval* such as a year, a quarter, or a month. Often this data is provided by a third party, such as the World Bank's World Development Indicators (WDI) or Systemic Peace's polity data; such providers are known in ICEWS parlance as a *data source*.

Each data source provides multiple types of data. For example, the World Bank's WDI has information about Gross Domestic Product (GDP), average life expectancy, per capita income, and thousands of other items. Each type of information is referred to as a *data parameter*. Though most data parameters used in the ICEWS system have a numeric *data type*, some data parameters may be temporal or textual in nature.

Putting together a country, a data parameter, and a time interval gives you a *data point*, which is the value of said parameter for said country at said point in time. For example, the WDI-provided GDP for China in the year 2011 was $11.3B USD. That number – 11.3 billion – would be a datapoint in the ICEWS system. How it is actually stored internally depends on its *scale*. For example, with a scale of 'US dollars', it would be stored as

11,290,910,568,573.3, while a scale of ‘millions of US dollars’ would store it as 11,290,910.6.

The following database tables are used to define state data in ICEWS:

Name	Purpose	When Populated
`datasources`	Holds metadata about the different data sources	By hand
`dataparameters`	Holds metadata about a data source’s published parameters	By custom-created data ingestion modules
`intervals`	Defines the time intervals at which data parameters may be published (e.g., yearly, monthly, etc.)	On database initalization
`parametertypes`	The types of data parameters: numeric, date, or string	On database initialization
`scales`	The scale needed to interpret the data (e.g., ‘millions of dollars’, ‘USD as of 2000’, etc.)	By custom-created data ingestion modules
`datapoints`	The value of a data parameter for a particular country at a specific point in time	By custom-created data ingestion modules
`datapoints_change_log`	Optional timestamped record of when existing datapoints values have changed	By custom-created data ingestion modules
`datapoints_change_ annotations`	Optional timestamped comments for when existing datapoints values have changed	By custom-created data ingestion modules

Table 10 - State Data-related Database Tables

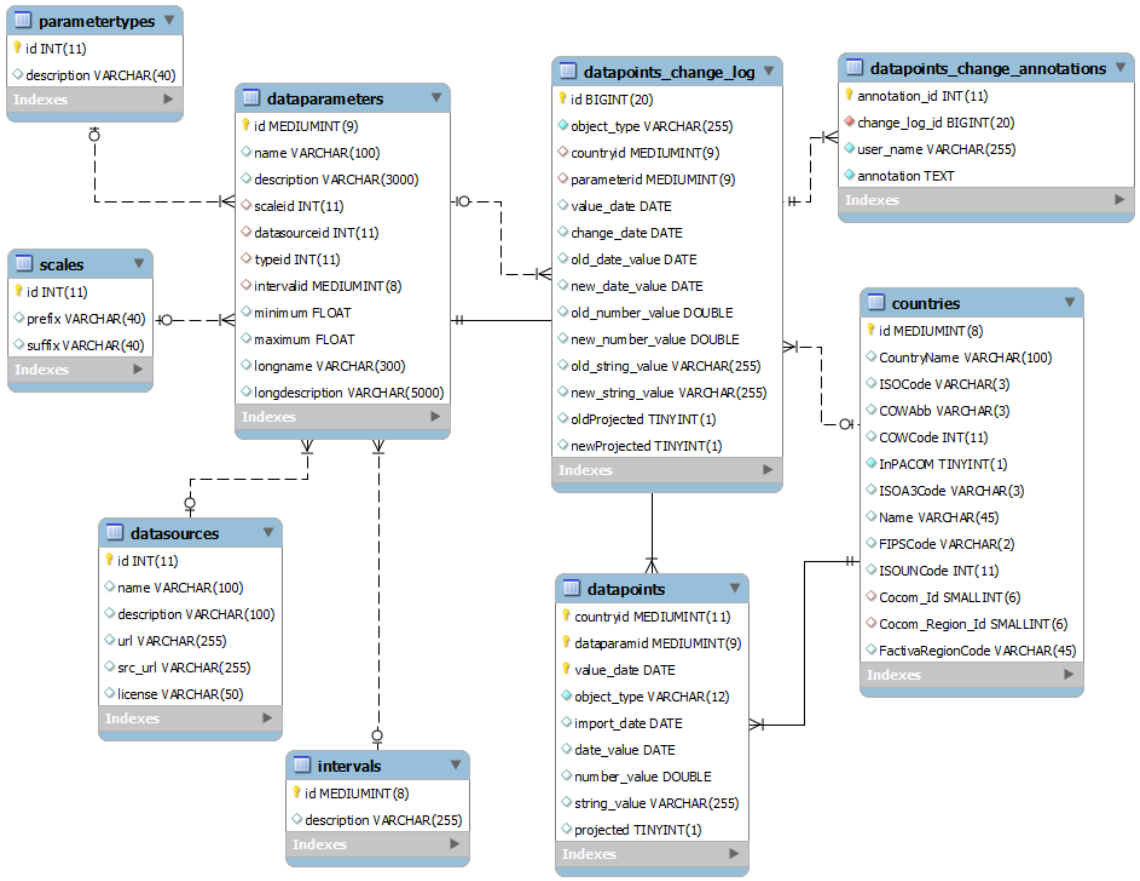

Figure 9 - State Data-related Database Tables

3.7 Other Tables

These are the tables that don't fit in elsewhere.

3.7.1 Event Browser Tables

There are a large number of 'transient' tables that are related to the event browser, used to store state while the event browser is in use. These include: `actorfilter`, `actorfilter_countries`, `actorfilter_dict_sectors`, `actorfilter_selections`, `dict_actorgroups`, `dict_actgrp_dict_actnodes`, `dict_actorgroups_dict_actors`, `eventdatafilter`, `eventtypefilter`, `eventtypefilter_eventtypes`, `eventgroupfilter`, `eventgroupfilter_eventtypes`, `publisherfilter`, `publisherfilter_selections`, `publishertypefilter`, and `publishertypefilter_selections`.

These tables only exist to support the event browser, and do not need to be initially populated.

3.7.2 ISPAN Security Tables

A number of tables exist for compatibility with ISPAN security classifications. Though this security information is not currently in use (everything is considered unclassified), it is pre-positioned for incorporation in an upcoming drop. These six tables start with 'b_' as the table name. You will need to refer to ISPAN documentation for a description.

Note that every ICEWS table in the G-ICEWS/ISPAN system has a field called `row_sec_lbl_id` to support the future incorporation of these tables.

3.7.3 Deprecated Tables

There are several tables that are currently deprecated, as they support ICEWS functionality that fell out of scope for Drop 15. These tables do still need to exist on the data serving machines, but they will not be accessed and their contents are undefined.

- **suggestedactors** is used by the Dictionary Lookup portlet's 'suggest actor' feature. This has been removed from Drop 15 as there is no CONOPS in place for reviewing suggested actors, though it remains in place in the W-ICEWS system at SOUTHCOM.

- **viewed_actors** is used by the event browser to record information about events that are viewed in the system. Originally this was used to validate the correctness of events, with the assumption being that if a user viewed the specific details related to an event but did not flag it as a bad event, it was likely a valid (correct) event. As there is no CONOPS to support reviewing this information, and as the ability to flag bad events has been removed, this functionality is no longer supported.

- **dict_actor_agent_mappings** is a table whose original purpose is shrouded in mystery, and is therefore deprecated.

- **dict_etcons_et_mappings** is a table once used as part of the aggregation code, but a redesign has made it obsolete.

- **gtds_annotations** is a table that once supported a user interface used to annotate information about the Ground Truth Data Set (GTDS) provided by ICEWS. It is no longer supported.

- **gtds_annotations_mappings** is an auxiliary table to the above, and also no longer supported.

Appendix A: Glossary

Actor. In an ***Event***, the group, individual, or location that is involved in the "*who*" or "*whom*" role. ***Actors*** may serve as the ***Source*** or ***Target*** of an ***Event***. In grammar terms, it would be a noun**.**

Actor Affiliation. An association between two actors, where one of the two actors must be a group or country. An individual who is a member of a group (e.g., Osama bin Laden as a member of Al Qaeda) would have an affiliation between themselves as an individual actor, and the group as a group actor.

Actor Dictionary. A set of XML files that define the ***Actors***, ***Agents, Event Types***, ***Sectors***, and ***Aggregation Queries*** in use in the ICEWS system.

Actor Membership. An alternate term for ***actor affiliation***.

Actor Pattern. A ***Pattern*** associated with a specific ***Actor***.

Actor Type. ***Actors*** are classified as one of four types: ***Individual***, ***Group***, ***Country***, or ***Location***.

Affiliation. See ***Country Affiliation*** or ***Sector Affiliation***.

Agent. A generic ***individual*** or ***group*** that can be combined with one of multiple ***country*** or ***country-derived actors*** into a ***composite agent-based actor***. Examples of agents include Fishermen, Military, or Democratic Party. In grammar terms, an agent is an improper noun.

Agent-Based Composite Actor. See ***Composite Agent-Based Actor***.

Agent Pattern. A ***Pattern*** associated with a specific ***Agent.***

Aggregating. Taking a ***country-affiliated filtered event set*** that corresponds to a particular ***time interval*** and calculating a numeric value based on it.

Aggregation. See **Event Aggregation**.

Aggregation Query. A specification for forming a ***filtered event set*** for aggregation purposes via a ***country constraint*** and zero or more ***sector constraints***.

CAMEO. Conflict and Mediation Event Observations. A coding scheme for event data developed initially at Kansas State University.

CAMEO Category. In the ***CAMEO*** coding scheme, ***CAMEO codes*** are organized into one of twenty high-level CAMEO categories.

CAMEO Code. In the ***CAMEO*** coding scheme, CAMEO codes are used as a short-hand method to identify types of events. There are 312 CAMEO codes in the CAMEO version that ***Event Types*** in ***ICEWS*** are based on, and they are arranged in twenty ***CAMEO Categories*.**

Category. See ***CAMEO Category.***

Composite Actor. An ***Actor*** that is not listed in the ***Actor Dictionary*** but is instead formed dynamically within the ***ICEWS*** system by combining a ***Sector*** with a ***Country*** to form a ***composite sector-based actor,*** or by combining an ***Agent*** with a ***named group*** or ***country actor*** to form a ***composite agent-based actor.***

Composite Agent-Based Actor. A ***composite actor*** that is formed dynamically by the ***ICEWS*** system by combining an ***agent*** with a ***group*** or ***country actor.***

Composite Sector-Based Actor. A ***composite actor*** that is formed dynamically by the ***ICEWS*** system by combining a ***sector*** with a ***country***.

Constraint. Also known as a ***criteria***, a restriction on some aspect of an event, such as its ***source*** or ***target***, for the purpose of generating a ***filtered event set***.

Country. A region that represents a distinct political entity, either an independent sovereign state or a non-sovereign state that is officially recognized by the US.

Country Affiliation. The country affiliation of an ***actor*** is the country (or, in some instances, the multiple countries) that the actor is closely affiliated with. For ***aggregations,*** the country affiliation is the affiliation of the ***source*** or ***target*** of all events in the ***filtered event set*** that underlies the aggregation; whether it is the source or target actor's affiliation is part of the aggregation definition.

Country-Based Constraint. A ***constraint*** placed on the ***target*** or ***source*** of an ***event*** based on the ***country affiliation*** of the ***actor*** in question.

Country-derived. An ***actor*** is said to be country-derived if it has a ***membership*** to a country (perhaps in context with a ***sector***), or with some ***actor*** that is in turn country-derived.

Criteria. Another term for a ***Constraint.***

Data Parameter. A particular type of data that a ***data source*** publishers, such as Gross Domestic Product (GDP) or average life expectancy.

Data Points. A value associated with a particular ***data parameter*** for a specific ***state*** and a specific ***time interval*** (e.g., the GDP of China in 2011).

Data Source. A typically third-party provider of ***state data***.

Data Type. How data may be expressed, in numeric, date, or string format.

Dictionary Editor. A graphical user interface that is used to edit the set of files making up an ***Actor Dictionary*.**

Dyadic Pair. The part of an ***aggregation query*** that specifies the ***source*** and ***target*** actors that are found in ***events*** of the ***filtered event set***.

Event. Information about *who* did *what* to *whom*, *when*, and *where*. The "*who*" and "*whom*" are the ***Source*** and ***Target*** respectively, the "*what*" is an ***Event Type*,** the "*when*" is a date, and the "*where*" is a ***Location*.**

Event Aggregation. The association of numerical data derived from event data with a given country for some time interval.

Event Intensity. A value ranging from -10 to +10, used to express the level of cooperation or hostility exhibited in the ***Event Type*** it is associated with. Cooperative events have a positive value while hostile events have a negative value, with values toward the edges of the range expressing a greater degree of hostility or cooperation and following a roughly linear scale for other values. Event intensities are expressed from a neutral point of view.

Event Query. See *Aggregation Query*.

Event Set. A collection of one or more *events* that are considered as a group.

Event Type. In an ***Event***, the "*what*" that occurs. It is an action that occurs between a ***Source*** and a ***Target***. In grammar terms, it would be an action verb.

Explicit (sector) membership. A ***sector membership*** that is explicitly and directly defined with the actor in question within the ***Actor Dictionary***.

Filtered Event Set. An ***event set*** that has been ***filtered*** such that it satisfies a set of ***constraints***.

Filtering. Applying a set of ***constraints*** to an ***event set***, typically as a precursor to ***aggregating*** them.

Goldstein value. An alternate term for ***Event Intensity***.

Group. An ***actor type*** that corresponds to multiple individuals with a common and publicly professed identity.

iCAST. The part of the ICEWS system that is concerned with forecasting aspects of country stability.

ICEWS. Integrated Crisis Early Warning System, an analytical software system developed by Lockheed Martin.

Implicit (sector) membership. A ***sector membership*** which is not specified explicitly in the ***Actor Dictionary***, but is instead inherited through the transitive nature of sector and actor memberships.

Individual. An ***actor type*** that corresponds to a single person.

Intensity value. An alternate term for ***Event Intensity***.

Internal Event. An ***event*** that occurs in the context of a single country, such that the ***country affiliations*** of the ***source*** and ***target*** resolve to the same country.

International Event. An ***event*** where the ***source*** and ***target country affiliations*** differ.

International Sector. See ***unaffiliated sector***.

Interval. See ***time interval.***

iTRACE. The part of the ICEWS system that is concerned with historical event trends.

Location. The "*where*" part of an ***event***. A location may be a country, province, district, city, location within a city, or region of multiple countries.

Membership. See Sector Membership.

Named Actor. A non-composite actor that is listed in the Actor Dictionary. Named actors must have a proper name used to identify them.

National Sector. 1. Most generally, a ***sector*** that is classified as existing only within the context of a ***country*** or ***country-derived actor***. 2. In ***event queries***, the national sector is a special sector that indicates the actor must be a country (i.e., have an ***actor type*** of country).

Paired Actor Reference. In a ***composite actor***, the ***actor*** with which the ***sector*** or ***agent*** is combined.

Paired Agent Reference. In a ***composite agent-based actor,*** the ***agent*** with which an ***actor*** is combined.

Pattern. A sequence of letters and underscores used to define how an actor or agent is represented in news stories. Though there are nuances beyond the scope of this document, they can be thought of as synonyms for an actor or agent, where spaces are

replaced with underscored and the case (e.g., uppercase, lowercase) of letters do not matter.[10]

Query. See ***Aggregation Query***.

SAE Code. Historically speaking, agents within the ICEWS system were based on an agent dictionary that had what were used as SAE Codes to represent them. These codes have been preserved for agents, for historical purposes.

Scale. For a ***data parameter***, the identifying scale in which it is expressed (e.g., billions of dollars, USD as of the year 2000, etc.).

Sector. A general role taken on by an actor or agent, sometimes in a time-constrained context. Examples of sectors include Government, Muslim, Dissident, and Refugees.

Sector Affiliation. An association between an ***actor*** or ***agent*** and a ***sector***. Sector affiliations may either be explicit, in which case they are defined in the ***Actor Dictionary***, or implicit, in which they are derived from the transitive nature of sector and actor affiliations.

Sector-Based Composite Actor. See *Composite Sector-Based Actor*

Sector-Based Constraint. A *constraint* placed on the *target* or *source* of an *event* based on the *sector affiliation* of the *actor* in question

Sector Membership. Used interchangeably with ***sector affiliation***.

Source. In an ***Event***, the ***Actor*** that is the instigator of the action.

State. See ***country***.

State Data. Typically numerical data that is associated with a particular ***country*** (or in geopolitical terms, a particular ***state***) for a given ***time interval*** such as a year, a quarter, or a month.

Target. In an ***Event***, the ***Actor*** that is the recipient of the action.

Time Interval. A unit of time, such as a month or a week. Time intervals may either be general (e.g., 'a week') or specific (e.g., 'the first week of 2006').

Unaffiliated Sector. A ***sector*** that is not constrained to the context of a specific country. Unaffiliated sectors are also known as ***international sectors*** due to their lack of country affiliation.

[10] ICEWS patterns are a close variant of TABARI patterns, as explained in the TABARI manual: http://eventdata.psu.edu/tabari.dir/tabari.manual.060228.pdf