

Luminosity Database: Status and Issues

Jim Linnemann

MSU

Taking Stock Meeting

June 11, 2007

The year in review

D0 reconstruction and CAF making using DB!

Calibration corrections done in production DB

Rename new columns today?

leave with extra columns until end p20 prod

Added view to Dbserver for Imaccess

about x 100 speedup

now within a factor of 2 (or better) wrt stage3

Dbserver support by S. White

Dbservers, and views, now fully in CVS

The Procedure (Executive Summary Version)

0. start new lumi dbserver for laccess (Luiz)
1. lmGrabber OFF (Vladimir Sirotenko)
2. full RMAN backup (DBA's)
3. get row counts of 3 affected tables (all up to present) (E)
4. add 8 new columns in 3 tables containing luminosity measurements (E)
new columns are named *_NEW
5. archiving OFF (DBA's)
Start lmGrabber version 2 (Vladimir Sirotenko)
6. define Oracle functions that deliver Period and corrected Luminosity (E)
they are used by the 3 scripts in the following step
7. run 3 scripts to populate 8 *_NEW columns (E/DBA's)
one script for each of the 3 affected tables (**guess: 1-6 days**)
faster on production db than on development: better hardware
DBA's may have to run scripts to insure complete logs
Don't run laccess now (to speed up scripts)
8. verify scripts completed execution on # rows expected (E)
9. archiving ON (DBA's)
10. turn OFF Luminosity db servers (DBA's)
so that errors won't occur for users while columns are renamed
Stop lmGrabber version 2 (Vladimir Sirotenko)
-  10. turn off farm and CAF production (hope a few hours) (Marco, Mike Diesburg)
11. rename 8 default luminosity columns to _OLD (E)
rename 8 *_NEW luminosity columns default columns names (E)
(Note: At this point, Operations Reports and lm_access will get corrected values from the production database)
12. lumGrabber version 3 ON (Vladimir Sirotenko)
13. turn ON Luminosity new db servers (DBAs, Marco)
Test, then turn on farm and CAF production (Marco, Mike Diesburg)
14. Testing begins
(Kayle runs ROOT plotting tests)
(Elizabeth runs row verification scripts)
15. when p20 reco production finishes
Delete _OLD columns (Igor Mandrichenko)
Go to final versions of dbserver (Marco, Luiz)

Pending Issues

Missing lbns?

Leave abandoned unless period F?

Verification lbn by lbn Vladimir M.

Understanding status codes Vladimir M.

Web reports plot issues resolved

code maintenance soon to go to Igor's group

Remove bad production checks

Decide on DB tags: explain in a web page

2a:E D05139.A5 48mb

D0 note 5139, appendix A5 has exact constants used

+ date tag in another field

Transition:

when do Igor et al take over db, web reports, etc?

Does Igor do column deletion, or Elizabeth?

Web page maintenance (Python issues)

Database evolution?

Delete _Old columns after p20 production done

Re-purpose 1-2 columns in lbn record

probably not much interaction with view

May remove as redundant some trigger-level status messages

We believe:

there are TOO MANY of them

we don't need them for actions

Still need verification of this

Verifying content: Im_access and python/sql jobs

This summer? Another recalibration of db?

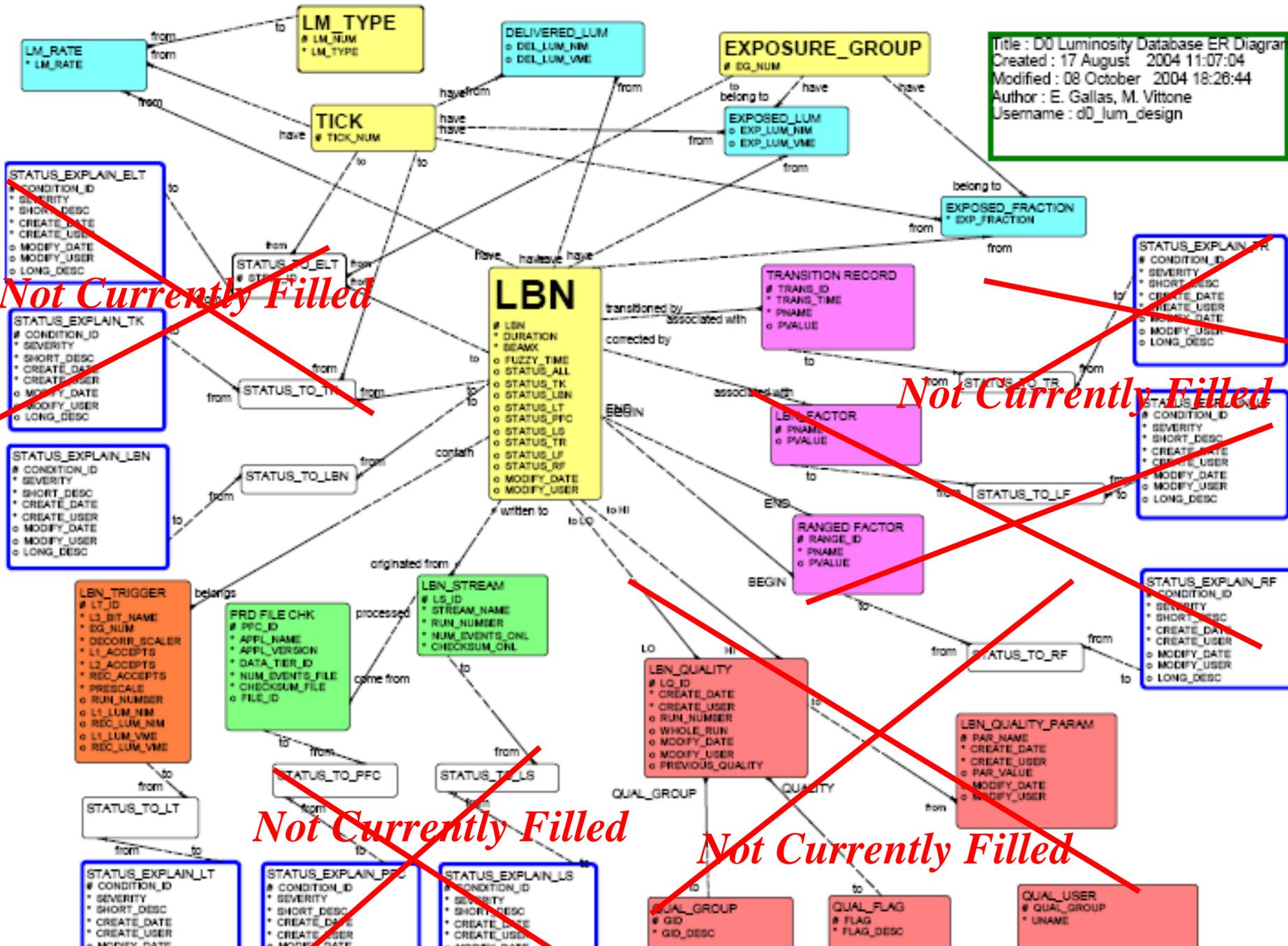
Title : D0 Luminosity Database ER Diagram
Created : 17 August 2004 11:07:04
Modified : 08 October 2004 18:26:44
Author : E. Gallas, M. Vittone
Username : d0_lum_design

~~Not Currently Filled~~

~~Not Currently Filled~~

~~Not Currently Filled~~

~~Not Currently Filled~~



Lm Access/DBserver plans post p20 production (end June?)

Working on multiple users

& requests for multiple luminosity types
delivered, triggered, recorded

Bad ticks (have 1st round code in place)

can't use cached "tick 0"=sum of ticks

must hand-sum over ticks, for lbnns at start of RunIIb

Considering whether need status at lbn_triggers level

Restrict access to stage3

then stop making new stage3 files

then kill them (end of summer?)

Dbserver reorganization

After current p20 production run

for now: temp servers looking at _old

rebuild servers with cxoracle

Merger Im_access and production farm servers

down to 2 instances?

change to C++?

step	column names	contents	read by	web/python	servers: dbsrv5 lmaccess	dbsrv4 reco farm	dbsrv5 caf maker	lmGrabber	plan step #
current status:									
old data	xx	old	all	sql/python	oracle	farm	user->oracle		
new data	xx	current	all	sql/python	oracle	farm	user->oracle	grab1	
start of recalibration:									
old data	xx	old	all	sql/python	lumi	farm	user->oracle		step 0 change needed
new data	xx	current	all	sql/python	lumi	farm	user->oracle	grab1	
during filling of new columns (after new column creation)									
old data	xx	old	all	sql/python	lumi	farm	user->oracle		step 5
old data	xx_new	corrected							
new data	xx	current	all	sql/python	lumi	farm	user->oracle	grab2	
new data	xx_new	current						grab2	
after column renaming									
old data	xx	corrected	lmaccess, web	sql/python	lumi				step 12, 13 stop and restart lumi
old data	xx_old	old	reco, caf			farm2	user->oracle2		
new data	xx	current	lmaccess, web	sql/python	lumi			grab3	
new data	xx_old	current	reco, caf			farm2	user->oracle2	grab3	
end production, after deletion:									
old data	xx	corrected	all	sql/python	lumi	farm3	user->oracle3		step 15
new data	xx	current	all	sql/python	lumi	farm3	user->oracle3	grab1	
	<i>xx=default names</i>			<i>use default name only</i>			<i>toggle: default/old</i>		

Dbrowsers:

farm:

a purely Python server now, with caching on

farm2 = farm, but using `_old`

perhaps farm3 becomes a C++ server with caching forwarding to a Python w/o caching
either "oracle3" or "lumi" code running on `dbsrv4`

user:

C++ server with caching on (different caching than farm)

forwards to non-caching Oracle python server on `dbsrv5` (currently "oracle", used for lumi)

user doesn't change; oracle changes instead

user and farm3 maybe can become the same code

oracle:

Python server with no caching

currently used for both lumi and forwarded to by `user(caf)`; later serving only `caf`, or replaced by lumi again

oracle2 accesses `_old`

oracle3 accesses default again; might become same as lumi, especially if load allows both on `dbsrv5`

lumi:

new separate python non caching server but with all code needed for `laccess`

might eventually run on either a `d0srv5`, or another machine, depending on load

might become same as oracle3

grab2

fills both default and `_new`

grab3

fills both default and `_old`
(at LEAST for delivered lum)

Production and testing

D0 and CD have different definitions of “production”

CD production is **reading** or writing prod DB

D0 production is

- official D0 software release

 - for general D0 users

- processing of millions of events with D0 official software release

Differing Consequences

Annoyance level: db recalibration and testing has interfered with other db verification: our problem

What consequences is dev testing meant to avoid?

D0: wrong calculation; halting production or bad performance

farm performance can only usefully be tested with production db

CD: wrong data writing, db structure problems, dbserver machine crashes/interference

Related: “whose” machines DBservers run on...