

Storing Event Data in a Relational Database

John S. Haggerty

November 4, 2008

Abstract

A schema for storing event data in a relational database is defined, implemented, and tested using events simulated by Pythia. SQL queries for some common analysis needs are explored.

1 Introduction

Attempts to store event data from high energy and nuclear physics experiments in databases have been disastrous. Yet relational databases are widely considered the most efficient way to store and access large amounts of data, and vast amounts of research have been done on them to make them scale to large sizes and to parallelize them, not to mention vast fortunes made from them.

Perhaps it is worth another attempt to use a relational database management system (RDBMS) to store and access physics data. In the past, high performance RDBMS's were an expensive commodity, licensed by node, with inaccessible documentation. Now, however, several databases (MySQL (<http://www.mysql.com/>), Drizzle (<https://launchpad.net/drizzle>), and PostgreSQL (<http://www.postgresql.org/>), among others) are open source, and are freely available. MySQL, recently acquired by Sun Microsystems, is particularly widespread, with an estimated eleven million installed servers.[1] Raw data seems to be a poor candidate for database storage, since it is basically binary data which needs to be processed into quantities which would be sensible to search, and the ability to efficiently search the data for particular events or tracks is the main value of using a database management system.

The purpose of this work is to determine how efficiently one might be able to store and retrieve large amounts of data in a relational database. It could serve some useful purpose in the future, for example by allowing students to access simulated data with a simple and familiar interface. It may also be useful to archive the results of detailed analysis of PHENIX data, like the best measured four vectors of leptons from J/ψ or W decay which would allow physicists outside the PHENIX collaboration to explore analyzed PHENIX data without the need to use the unique PHENIX analysis environment but it is mainly a research project in data access techniques.

2 Database Design

The database, which we call “pythiadb” is divided up into three kinds of tables, one which describes individual Pythia runs, one with a simple description of the most general properties of the event, indexed by the run number and the event number, and one which have descriptions of the tracks. This database was constructed with all kinds of events in the same database, so a query seeking a particular kind of event would select on run number (runno) to select a particular process. Of

course, one could create separate databases which might have different kinds of events, or different event generators, but the table structure described here has proven to be adequate to select the desired events.

No great attempt was made to optimize the database structure, but the tables were indexed on run number (runno) and event number (evno) which speeds up queries such as the example invariant mass queries described later, which could be clearly seen with the “EXPLAIN” command. The default (for the MySQL 4.1.22 server used) database engine was MyISAM.

At the time of this note, there are 13 runs of pp collisions with 500 GeV center of mass energy with 100,200 events, and 22,912,581 tracks (Pythia records many intermediate state particles as well as final state particles). The database directory is 4.6 Gbyte in size on the server, which corresponds to about 48 kbyte/event or about 214 bytes/track. There are 186 bytes/track in the definition of the pythiatrack table bytes/track, so there is about 15% overhead from the indexes and pythiaevent table.

pythiarun The run table has one row for every Pythia run.

pythiaevent The event table has one row for every PHENIX event. The table has a composite unique primary key (runnumber, eventnumber) and contains some basic information which summarizes the nature of the event (the trigger type, for example).

pythiatrack The track table describes the properties of every track in the event.

A program called mpythia was written to take the Pythia variables for each run, event, and track, and store it in the database. In addition to the minimal event record described in the Pythia documentation[3], some derived quantities were stored in the database as well (pseudorapidity, and p_T , for example) both to allow checks for internal consistency and to simplify some of the queries that can be easily envisioned.

3 Queries

- A simple query:

```
select -ln(tan(theta/2)),eta from pythiatrack where abs(id)=11
```

- Invariant mass:

```
select sqrt(pow(a.e+b.e,2)-pow(a.px+b.px,2)-pow(a.py+b.py,2)-pow(a.pz+b.pz,2))
as mab from pythiatrack a,pythiatrack b
where a.runno=7 and b.runno=7 and a.evno=b.evno
and abs(a.id)=11 and abs(b.id)=11 and a.id+b.id=0
```

- Find energy of e^\pm which come from the decay of a W^\pm (d is the “daughter” row and m is the “mother” row). The result of the query is plotted in Figure 1.

```
select d.e from pythiatrack m, pythiatrack d
where m.runno=12 and m.runno=d.runno and m.evno=d.evno
and abs(d.id)=11 and abs(d.eta)<0.35 and (abs(d.phi)<0.78 or abs(d.phi)>2.36)
and abs(m.id)=24 and d.mother1=m.partno
```

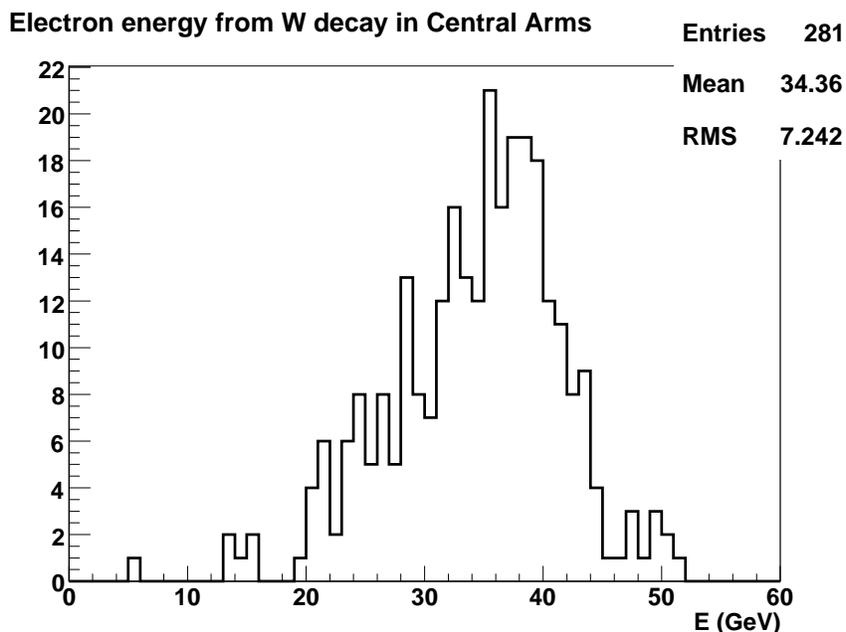


Figure 1: Electron energy from W decays as found from the query described in the text.

The last query (shown in Figure 1) is the most complex, and it took about a minute (mysql reported 66.53 sec).

4 Tools

A few simple tools were developed to execute queries and make plots of the results. Instead of the canonical method used by event driven analyses, in which a block of data is identified as an event, and then all or part of the data are extracted and unpacked into local variables, an attempt was made to use the SQL query itself to locate the desired data.

4.1 ROOT

A simple macro which connects to the database, makes a query, and plots the first returned field from the database was written. Figure 1 is an example of a plot made with this Root macro.

4.2 Dynamic Web Pages

A simple web interface to the data was developed in PHP using the JpGraph[2] plotting package. This allows some quick looks at the data, and allows some queries to be made from web forms. The web page <http://logbook.phenix.bnl.gov/pythiadb> is the top level page. the run index page

Pythia Runs

Connected successfully to phnxdb1
Page updated: 2008-10-24 14:28:22

Create histogram

Create scatterplot

Run	idA	idB	eCM	nTried	nAccepted	sigmaGen	sigmaErr
0	2212	2212	500	5000	5000	59.4422	5.66498e-08
1	2212	2212	500	22408	5000	191.262	1.53891
2	2212	2212	500	24353	5000	0.00796779	5.74022e-05
3	2212	2212	500	24932	5000	0.00125979	8.93272e-06
4	2212	2212	500	24715	5000	3.34024e-06	2.6798e-08
5	2212	2212	500	771717	5000	2.08769e-11	9.1682e-14
6	2212	2212	500	126509	5000	5.69369e-06	3.6549e-08
7	2212	2212	500	35735	5000	0.0726807	0.000525301
8	2212	2212	500	5000	5000	59.4422	5.66498e-08
9	2212	2212	500	30925	5000	0.000410799	3.27718e-06
10	2212	2212	500	460	100	3.30377e-06	1.95495e-07
11	2212	2212	500	3403	100	2.08377e-11	8.84669e-13
12	2212	2212	500	247339	50000	3.34099e-06	8.46133e-09

[About Pythia data](#)

Figure 2: Index of Pythia runs.

is show in Figure 2. The run numbers are linked to the log files from the Pythia runs so that one can examine what went into the runs.

Data can be explored graphically with the histograms and scatterplots which can be constructed from SQL queries. An example of the menu is shown in Figure 3, where the column names in each table are listed. Figure 4 is the JpGraph equivalent of the Root plot shown in Figure 1.

5 Conclusion

Relational databases provide a very uniform way to access data, and enforces strong typing of data. It provides an ideal framework for sharing data without sharing the software used to create it, and thus decouples the analysis of the data from the software used to insert the data in the database. It can potentially provide a conduit for exporting analyzed data to users who might not be fully aware of the details of the analysis used to produce it. Although specific queries can be optimized by the addition of indexes, there is no optimization that will make all queries equally efficient. For simple data exploration or transport, the database provides a well understood framework.

References

- [1] Charles Babcock. Sun locks up mysql, looks to future web development. http://www.informationweek.com/news/software/open_source/showArticle.jhtml?

Create Histogram

Connected successfully to phnxdb1

Columns in table pythiarun

runno	idA	idB	code	nTried	nAccepted	sigmaGen	sigmaErr
-------	-----	-----	------	--------	-----------	----------	----------

Columns in table pythiaevent

runno	evno	size	idA	idB	pzA	pzB	cA	cB	mA
mB	codeCM	s	code	nFinal	isResolved	isDiffractionA	isDiffractionB	isMinBias	isLHA
atEndOfFile	hasSub	codeSub	nFinalSub	id1	id2	x1	x2	y	tau
pdf1	pdf2	QFac	Q2Fac	isValence1	isValence2	alphaS	alphaEM	QRen	Q2Ren
mHat	sHat	tHat	uHat	pTHat	pT2Hat	m3Hat	m4Hat	thetaHat	phiHat
nTried	nSelected	nAccepted	sigmaGen	sigmaErr	bMI	enhanceMI	pTmaxMI	pTmaxISR	pTmaxFSR
nMI	nISR	nFSRinProc	nFSRinRes	errorTotalNumber					

Columns in table pythiatrack

i	runno	evno	partno	id	status	mother1	mother2	daughter1	daughter2
col	acol	px	py	pz	e	m	scale	hasVertex	sProd
yProd	zProd	iprod	tau	isFinal	pT	mT	theta	phi	ly
eta									

Query:

select d.a from pythiatrack m, pythiatrack d where m.runno=12 and m.runno=d.runno and m.evno=d.evno and abs(d.id)=11 and abs(d.eta)<0.35 and (abs(d.phi)<0.78 or abs(d.phi)>2.36) and abs(m.id)=24 and d.l
 Number of bins: 60 Minimum: 0 Maximum: 60.0

[About Pythia data](#)

Figure 3: Creating an SQL query which will scatterplot the two fields returned by the query.

articleID=206900327, February 26, 2008.

[2] Aditus Consulting. Jpgraph. <http://www.aditus.nu/jpgraph/>, June 15, 2008.

[3] Torbjorn Sjostrand, Stephen Mrenna, and Peter Skands. A brief introduction to pythia 8.1. <http://home.thep.lu.se/~torbjorn/Pythia.html>, October 2007.

e

query: select d.e from pythiatrack m, pythiatrack d where m.runno=12 and m.runno=d.runno and m.evno=d.evno and abs(d.id)=11 and abs(d.eta)<0.35 and (abs(d.phi)<0.78 or abs(d.phi)>2.36) and abs(m.id)=24 and d.motherl=m.parno
bins: 60 xmin: 0.0 xmax: 60.0
rows returned: 281
underflows: 0 entries: 281 overflows: 0
minimum: 5.246620 maximum 51.784900
average of entries: 34.361990

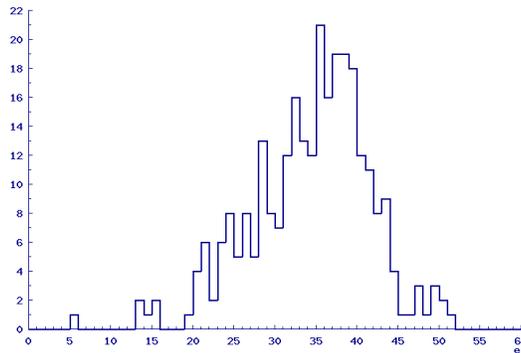


Figure 4: The plot returned by the query shown in the text.