Discovering Datascope

A Relational Database Management System

Daniel Quinlan
BRTT, Inc.

Database?

- Most collections of information could be viewed as a database, eg my desk.
- more organization: collection of waveform files
- more constraints: formal databases
- relational is the most popular formal database organization

Relational?

- very simple
  - tables of records
  - records have fixed set of fields
  - fields each have single datum
- A table is a "set", not a list: order is unimportant.
- Every record in a table is unique.

fields+tables = schema

- CSS3.0
  Center for Seismic Studies, version 3.0
  - site/sitechan/sensor/instrument
  - wfdisc
  - arrival/assoc/origin/event

One place

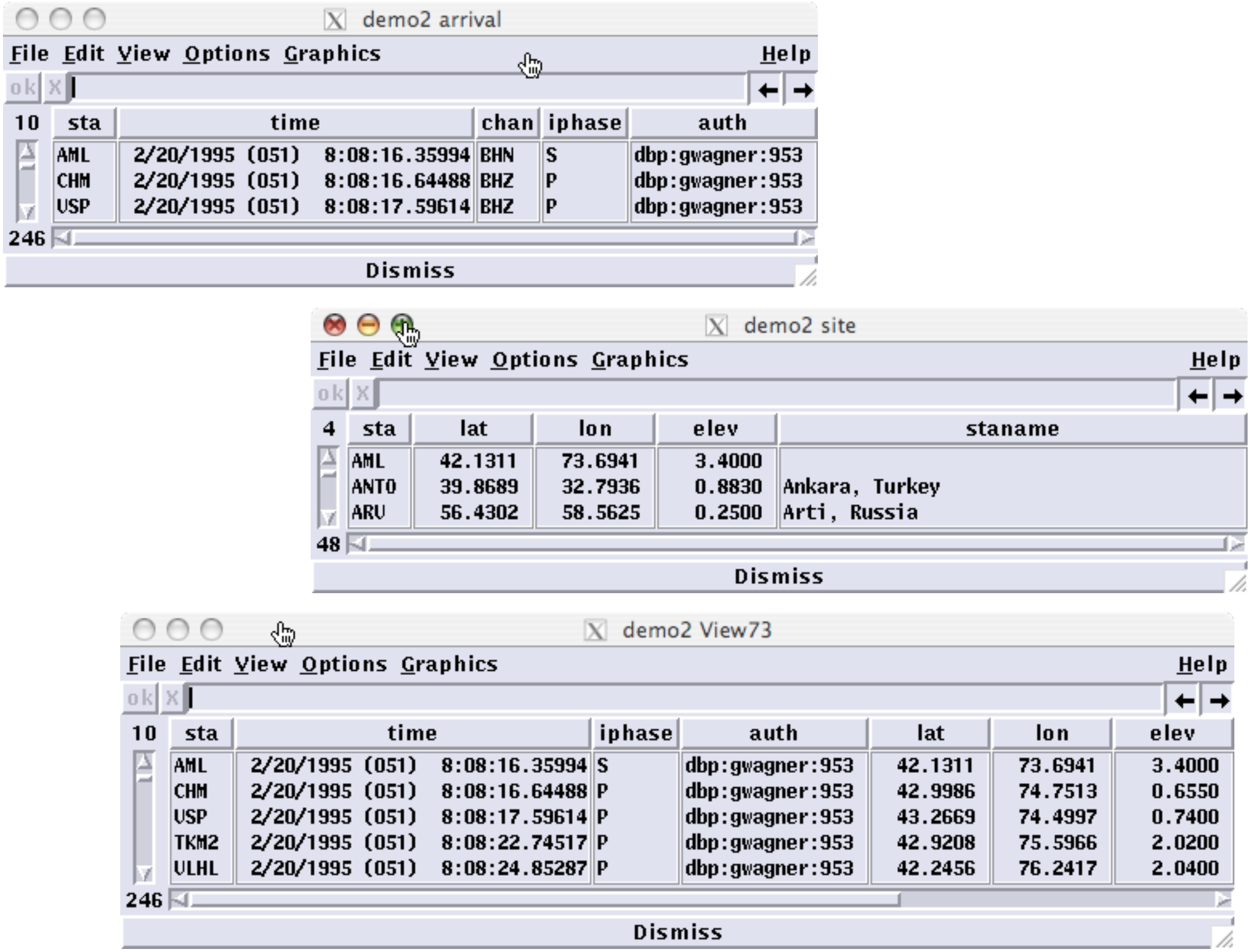
- A piece of information is found in only one place.
  - lat/lon in site table
  - response in instrument table

Most important Operations

- sort
- subset
- join

What is a join?

- for example, given an arrival pick, need to know the location of station to use in event location
- ie, given station name and arrival time, lookup the location of the station in the site table.
- consider new virtual table which combines arrival and site table; each record has 1 arrival plus matching record from site table


demo2 arrival
File Edit View Options Graphics Help
ok X
10
sta time chan iphase auth
AML 2/20/1995 (051) 8:08:16.35994 BHN S dbp:gwagner:953
CHM 2/20/1995 (051) 8:08:16.64488 BHZ P dbp:gwagner:953
USP 2/20/1995 (051) 8:08:17.59614 BHZ P dbp:gwagner:953
246
Dismiss
demo2 site
File Edit View Options Graphics Help
ok X
4
sta lat lon elev staname
AML 42.1311 73.6941 3.4000
ANTO 39.8689 32.7936 0.8830 Ankara, Turkey
ARU 56.4302 58.5625 0.2500 Arti, Russia
48
Dismiss
demo2 View73
File Edit View Options Graphics Help
ok X
10
sta time iphase auth lat lon elev
AML 2/20/1995 (051) 8:08:16.35994 S dbp:gwagner:953 42.1311 73.6941 3.4000
CHM 2/20/1995 (051) 8:08:16.64488 P dbp:gwagner:953 42.9986 74.7513 0.6550
USP 2/20/1995 (051) 8:08:17.59614 P dbp:gwagner:953 43.2669 74.4997 0.7400
TKM2 2/20/1995 (051) 8:08:22.74517 P dbp:gwagner:953 42.9208 75.5966 2.0200
ULHL 2/20/1995 (051) 8:08:24.85287 P dbp:gwagner:953 42.2456 76.2417 2.0400
246
Dismiss

Key Datascope concepts

- schema is relatively static file describing fields and tables in considerable detail.
  - creating database implicitly creates tables
- tables are plain ascii fixed format files
- Datascope is essentially a library of routines implementing operations on a database, with representations in a variety of languages:

  c, perl, tcl/tk, shell, matlab, php, python(?)

schema fields

- fields are same across all tables in schema

```
Attribute lat
        Real (9)
        Format ( "%9.4f" )
        Null ( "-999.0000" )
        Range ( "lat >= -90.0 && lat <= 90.0" )
        Units ( "Degrees" )
        Description ( "estimated latitude" )
        Detail {
                This attribute is the geographic
                latitude.  Locations
                north of the equator have
                positive latitudes.
        }
        ;
```

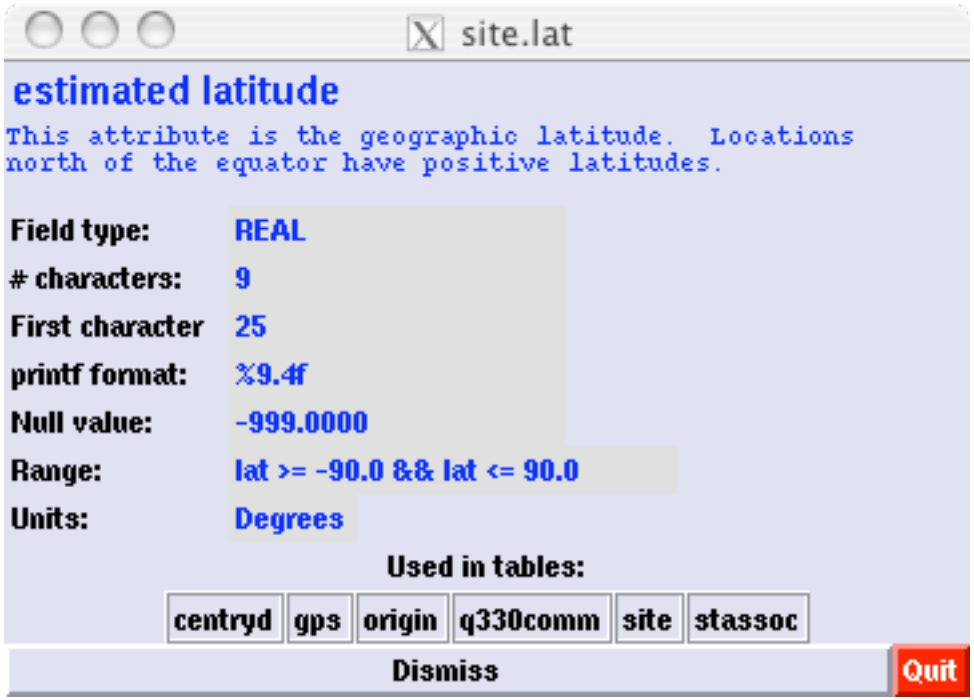
- more information

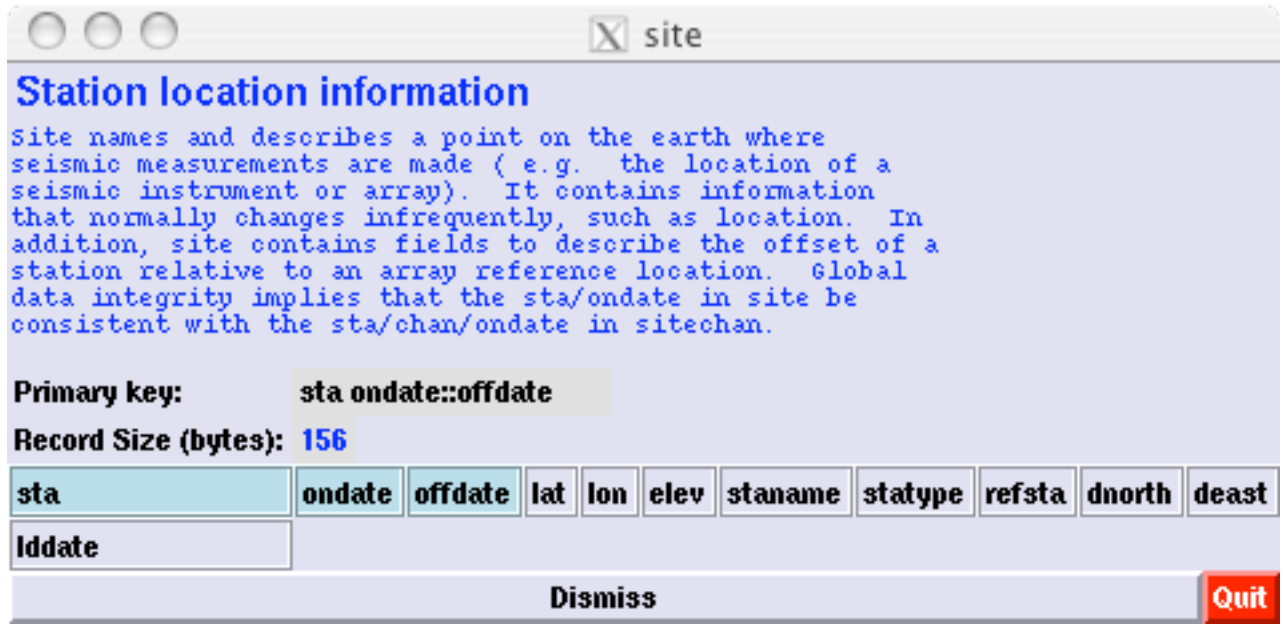
schema tables

```
Relation site
    Fields ( sta ondate offdate lat lon elev staname
           statype refsta dnorth deast lddate )
    Primary ( sta ondate::offdate )
    Description ( "Station location information" )
    Detail {
        Site names and describes a point on the earth
        where seismic measurements are made (e.g. the
        location of a seismic instrument or array).
        It contains information that normally changes
        infrequently, such as location.  In addition,
        site contains fields to describe the offset of
        a station relative to an array reference
        location.  Global data integrity implies that
        the sta/ondate in site be consistent with the
        sta/chan/ondate in sitechan.
    }
    ;
```

- note primary key information

dbhelp


site.lat
estimated latitude
This attribute is the geographic latitude. Locations north of the equator have positive latitudes.
Field type: REAL
characters: 9
First character 25
printf format: %9.4f
Null value: -999.0000
Range: lat >= -90.0 && lat <= 90.0
Units: Degrees
Used in tables:
centryd gps origin q330comm site stassoc
Dismiss
Quit


site
Station location information
Site names and describes a point on the earth where seismic measurements are made (e.g. the location of a seismic instrument or array). It contains information that normally changes infrequently, such as location. In addition, site contains fields to describe the offset of a station relative to an array reference location. Global data integrity implies that the sta/ondate in site be consistent with the sta/chan/ondate in sitechan.
Primary key: sta ondate::offdate
Record Size (bytes): 156
sta ondate offdate lat lon elev staname statype refsta dnorth deast
lddate
Dismiss
Quit

relational operations

- subset

  dbsubset db.site 'lat > 45'

- sort

  dbsort db.site sta

- join

  dbjoin db.arrival site

project/select

- dbselect - sta lat lon elev staname

```
% dbsubset $db2.site 'lat > 45' | \
  dbsort - sta | \
  dbjoin - arrival | \
  dbselect - sta lat lon arid arrival.time chan
AKT  50.4348   58.0167    14364   793267845.36933 BHZ
AKT  50.4348   58.0167    14373   793267990.83460 BHN
AKT  50.4348   58.0167    14377   793268086.67671 BHN
ARU  56.4302   58.5625    14366   793267899.68275 BHN
```

Keys

- primary key:
  some subset of fields in table which uniquely identify a record
- alternate key:
  typically a shorthand "id" field
  - eg, in arrival

    sta time (physical meaning)

    arid (alternate key, shorthand id)

join

- generally, find records from two tables where some condition is met
- for arrival and site, we want
  - sta codes same
  - arrival time matches range ondate::offdate

natural join

- Datascope infers the join keys based on key names and some heuristic rules

```
% dbjoin -v $db2.arrival site > /dev/null
Beginning with demo2.arrival
   joining to table site with keys:
        sta
        time == ondate::offdate
    result has 246 records
```

- for each record in arrival, find record(s) in site which match sta, time inside ondate::offdate

ranges

- time ranges are important in various tables
- join keys over ranges are complex, eg wfdisc-site:

```
(time > ondate and time < offdate)
or (ondate > time and ondate < endtime)
```

- conversion required

dbe demo

- dbe $db2
- open origin
- arrange
- record view
- time formatting
- map
- zoom out/in
- projections

- search mb > 2
- subset mb > 2
- open site
- sort by sta
- sort by distance
- could run map
- open arrival
- join to site (show join keys)
- graph amp vs per, show log/log
- graph amp vs time
- show saving text

- open instrument
- show response
- open response file
- open wfdisc
- show trace
- open origin
- show associated waveforms

~/.dbe.pf

```
graphics        &Arr{
    wfdisc      &Tbl{
        Waveforms trdisp -
#       filenames  dbselect - extfile() | xargs ls -l
    }
    instrument  Response  dberesp -
    stage       Response  dberesp -
    site        Map       dbmap_gui -
    origin      Map       dbmap_gui -
    origin      Waveforms origin_display -
    }
```

```
@db = dbopen_table ( $db, "r" ) ;
$db[3] = 0 ;
eval {
   ($prefor) = dbgetv(@db, qw(prefor)) ;
} ;
if ( ! $@ ) {
    @dborigin = dblookup (@db, 0, "origin", 0, 0) ;
    @db = dbjoin (@db, @dborigin) ;
    @db = dbsubset(@db, "prefor==orid" ) ;
}
$n = dbquery(@db, dbRECORD_COUNT) ;

if ( $n < 1 ) {
    print STDERR "no origin found\n" ;
    exit 1 ;
}

@dbassoc = dblookup (@db, 0, "assoc", 0, 0) ;
@db = dbjoin (@db, @dbassoc) ;
@dbarrival = dblookup (@db, 0, "arrival", 0, 0) ;
@db = dbjoin (@db, @dbarrival) ;
```

```
$max = dbex_eval(@db, "min(arrival.time)") ;
$db[3] = 0 ;
($min) = dbgetv(@db, qw(origin.time)) ;
$min -= 10 ;
$max += 30 ;

$n = dbquery(@db, dbRECORD_COUNT) ;
for ($db[3] = 0 ; $db[3] < $n ; $db[3]++ ) {
    $sta = dbgetv(@db, qw(sta)) ;
    push(@sta, $sta) ;
}
$subset = sprintf("sta =~ /%s/", join('|', @sta)) ;

$dbname = dbquery(@db, dbDATABASE_NAME) ;
$cmd = "trdisp -s '$subset' $dbname" ;
# print STDERR "$cmd\n" ;
system ( "$cmd &" ) ;
```

database integrity

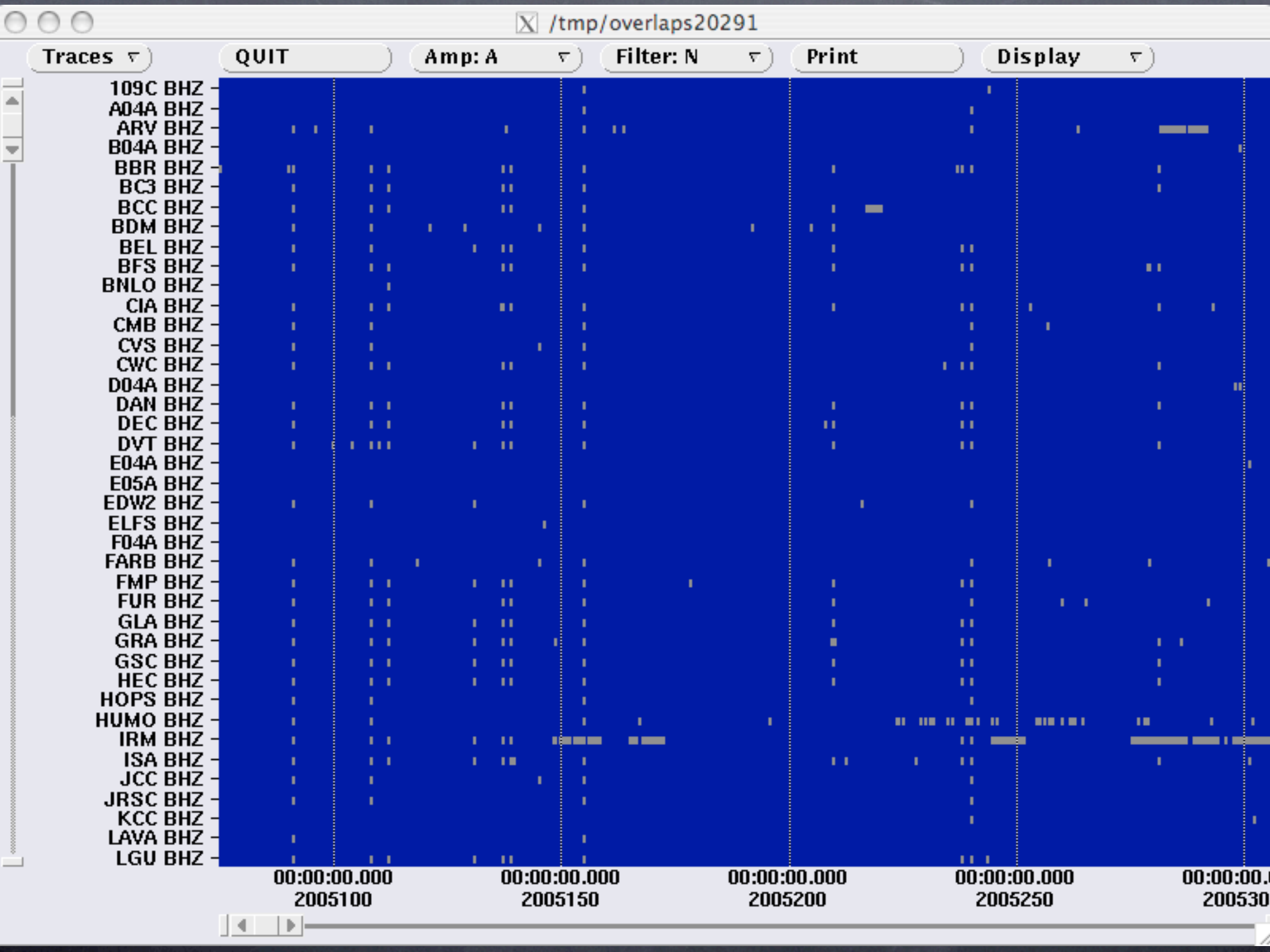
- dbverify
  - field ranges
  - uniqueness
  - referential integrity for ids
  - external file existence
  - other tests

example:

- Use dbverify to examine overlaps in wfdisc

```
dbverify -tk abc.wfdisc
Keys for records #2528 and #2529 in table wfdisc match:
  sta           AAK                          | AAK
  chan          BHE                          | BHE
  time           4/07/1999  23:38:50.000     |  4/07/1999  23:38:54.000
  endtime        4/07/1999  23:38:54.975     |  4/07/1999  23:38:58.975
Keys for records #2530 and #2531 in table wfdisc match:
  sta           AAK                          | AAK
  chan          BHE                          | BHE
  time           4/07/1999  23:39:00.000     |  4/07/1999  23:39:06.000
  endtime        4/07/1999  23:39:09.975     |  4/07/1999  23:59:59.975
```

```
open ( INPUT, "dbverify -tk $input |" ) ;
$overlaps = "/tmp/overlaps$$" ;
@db = dbopen ( $overlaps, "r+" ) ;
@db = dblookup ( @db, 0, "wfdisc", 0, 0 ) ;
dbtruncate ( @db, 0 ) ;
while ( <INPUT> ) {
    if ( /^  time/ ) {
        ($lbl, $time_dt1, $time_t1, $sep, $time_dt2, $time_t2) =
            split ( ' ' ) ;
    } elsif ( /^  endtime/ && $chan =~ /.*Z.*/ )  {
        ($lbl, $endtime_dt1, $endtime_t1, $sep, $endtime_dt2,
            $endtime_t2) = split ( ' ' ) ;
        $time = &max( str2epoch ( "$time_dt1 $time_t1" ),
                      str2epoch ( "$time_dt2 $time_t2" )) ;
        $endtime = &min( str2epoch("$endtime_dt1 $endtime_t1"),
                         str2epoch("$endtime_dt2 $endtime_t2")) ;
        $db[3] = dbaddnull (@db) ;
        $nsamp = ($endtime-$time);
        $result = dbputv ( @db, "sta", $sta, "chan", $chan,
                "time", $time, "endtime", $endtime,
                "nsamp", $nsamp, "samprate", 1 ) ;
    } elsif ( /^  sta/ )  {
        ($lbl, $sta ) = split ( ' ' ) ;
    } elsif ( /^  chan/ )  {
        ($lbl, $chan ) = split ( ' ' ) ;
    }
}
system ( "trdisp $overlaps" ) ;
```


/tmp/overlaps20291
Traces
QUIT
Amp: A
Filter: N
Print
Display
109C BHZ
A04A BHZ
ARV BHZ
B04A BHZ
BBR BHZ
BC3 BHZ
BCC BHZ
BDM BHZ
BEL BHZ
BFS BHZ
BNLO BHZ
CIA BHZ
CMB BHZ
CVS BHZ
CWC BHZ
D04A BHZ
DAN BHZ
DEC BHZ
DVT BHZ
E04A BHZ
E05A BHZ
EDW2 BHZ
ELFS BHZ
F04A BHZ
FARB BHZ
FMP BHZ
FUR BHZ
GLA BHZ
GRA BHZ
GSC BHZ
HEC BHZ
HOPS BHZ
HUMO BHZ
IRM BHZ
ISA BHZ
JCC BHZ
JRSC BHZ
KCC BHZ
LAVA BHZ
LGU BHZ
00:00:00.000
2005100
00:00:00.000
2005150
00:00:00.000
2005200
00:00:00.000
2005250

Other relational operations

- dbtheta
  arbitrary join
- complementary anti-join operations
  - dbsever -- eliminate one table from join
  - dbseparate -- separate records for one table of view
- dbnojoin
  records from table which don't join
- dbgroup
  group adjacent records from sorted view with matching fields

Command line operations

- dbdiff
  compare two databases or tables
- dbcp
  copy a database or table
- dbset
  change value (often a key) globally
- dbunjoin
  create new database from view

More commands

- dbaddv -i
  add records to a table from command line, or script
- dbdesign
  edit a schema
- dbcalc
  utility for evaluating expressions

expressions

- sin, cos, tan, atan, log, exp, floor, ceil, min, ...
- time conversion
- seismic travel time
- spherical geometry: distance, azimuth
- regular expressions
- concatpaths(a,b), dirname, basename,
- strlen(), substr(), null("field-name")
- execute command with ["wc", extfile()]

What is a view?

- a table of database pointers, each pointer identifying a single record from a "base" table
- all fields for base tables in view are present
- one exception: in a grouped view, some of the fields are represented directly in the view, and there is always a "bundle" pointer which identifies a range of records in another view.

Why not SQL?

- First problem: convince seismologists to use relational database (problem still not solved)
- Easy to understand
- Easy to use
- SQL tends to hide rather than illuminate
- more expressions and operations
- historically, other options limited, expensive eg, Oracle, Sybase