The Design and Implementation of the Pavement Research Center Heavy Vehicle Simulator Database

Report to the California Department of Transportation

Prepared by
Jeremy Lea and Lorina Popescu

February, 2003

University of California
Institute of Transportation Studies
Pavement Research Center

TABLE OF CONTENTS

1.0 INTRODUCTION

This document covers the design, implementation and usage of the heavy vehicle simulator (HVS) database in use by the Pavement Research Center (PRC) at the University of California (UC), which is referred to as the PRC-HVS database. This report describes the portion of the database used to store all of the data collected directly from HVS test sections. The same database is used to store data from both flexible and rigid pavement sections. Not included in this report are discussions of the environmental data, long term pavement performance data, and laboratory testing data, which are also included in the PRC database and which have been collected since 1995 during the operation of the two Caltrans heavy vehicle simulators.

This report is broken into three main sections. The first provides some background about the HVS activities and the reasons for various choices in the database design. The second covers the design of the database. The final section covers the processing of the data as it is loaded into the database.

Besides the raw data files collected during testing, the database consists of two components: a Microsoft Access® database, `PRC-HVS.mdb` (referred to as the database) and a Microsoft Excel® spreadsheet, `PRC-HVSLoader.xls`, (referred to as the loader). The database is also stored in an Oracle® database, which is used to drive a World Wide Web site. The Web site can be used to view the data.

2.0 BACKGROUND

The HVS was developed in South Africa, by the CSIR, and used there mainly for the testing of thin asphalt concrete pavements. The modes of distress on South African pavements tended to be permanent deformation in the various layers and surface cracking, and so the

instrumentation traditionally used on HVS test sections tries to capture these failure modes and their causes. The instruments used included the Multi-Depth Deflectometer (MDD), the Road Surface Deflectometer (RSD), the Profilometer and the Crack Activity Meter (CAM). The data acquisition system (DAS) used on the early HVS tests were some of the first microcomputers used in South Africa, and very advanced for their time. Because of limited memory and speed the number of data points collected for each of these instruments was limited to 256 points (i.e., a one byte counter). While the DAS has been continuously upgraded over the years, the file formats and this limitation have not been changed because there was no need for increased data resolution.

The HVS database designed and still used by the CSIR was designed to accommodate the testing style and instrumentation of these sections. When the two Caltrans HVS's began work in California, these file formats and testing procedures were adopted. However, as the Cal/APT project progressed, various other instruments have been added and the testing procedures altered in various ways. This was done to better capture the response and distress of the thick AC pavements and rigid pavements used in California.

For rigid pavements, a number of new instruments were used, including strain gauges and the Joint Deflection Measuring Device (JDMD). In addition, the testing protocols were altered from those used for flexible pavements. This was done to better capture the behavior of rigid pavements, since they exhibit deflections even when not being trafficked.

In 1999 a new data acquisition system was delivered to the PRC, specifically designed for rigid pavement testing. With this system, the instrumentation can be read while the machine is trafficking at full speed, instead of just at creep speed as with the old system. The output files

from this system differ from the outputs of the older data acquisition systems. This new system is now also installed on the upgraded HVS used primarily for flexible pavement testing.

One objective of the new database structure and its integration with recent changes in the HVS instrumentation, electronics, and operation of the two HVS's is to make both testing frames and data acquisition systems identical and interchangeable.

2.1 Section Location Coordinate System

With the old DAS, each type of instrument was recorded one at a time. For each instrument, 256 data points were recorded. Normally three repeats were performed and the data for each instrument type was stored in one file, with a header that listed information such as the section number, the date, wheel load, and which instrument was being read.

HVS test sections are typically eight meters long and one meter wide, and marked with station numbers from 0 to 16 down one side. This side was referred to as the *caravan side* (CS) because it was the side where the data collection caravan (trailer) was located. The other side of the section was known as the *traffic side* (TS), because it was normally on the side of the machine where the traffic from whichever road is being tested was running. The HVS is normally towed onto site, so the tow end (TE) is normally on the right and the cabin end (CE) on the left when looking at the section from the caravan side.[1]

Most instruments were located on the center line (CL) of the section. If an instrument was located elsewhere on the section, for example, at station 8 near at the traffic side (TS), its position was recorded accordingly (8TS). This method of orientation is shown in Figure 1.

[1] This is referenced to right-hand drive. The ends are reversed in South Africa where traffic flows on the left-hand side of the road.

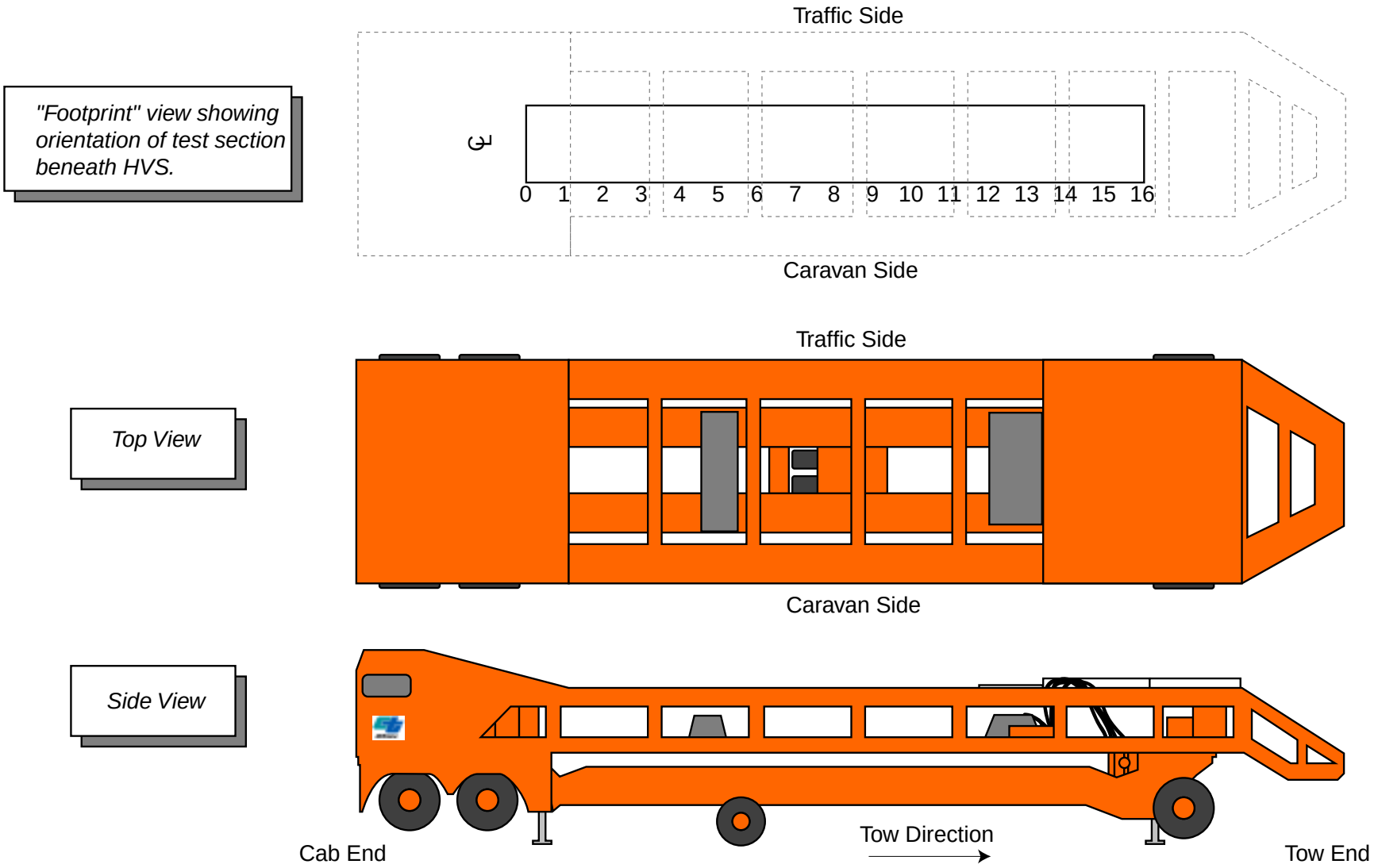
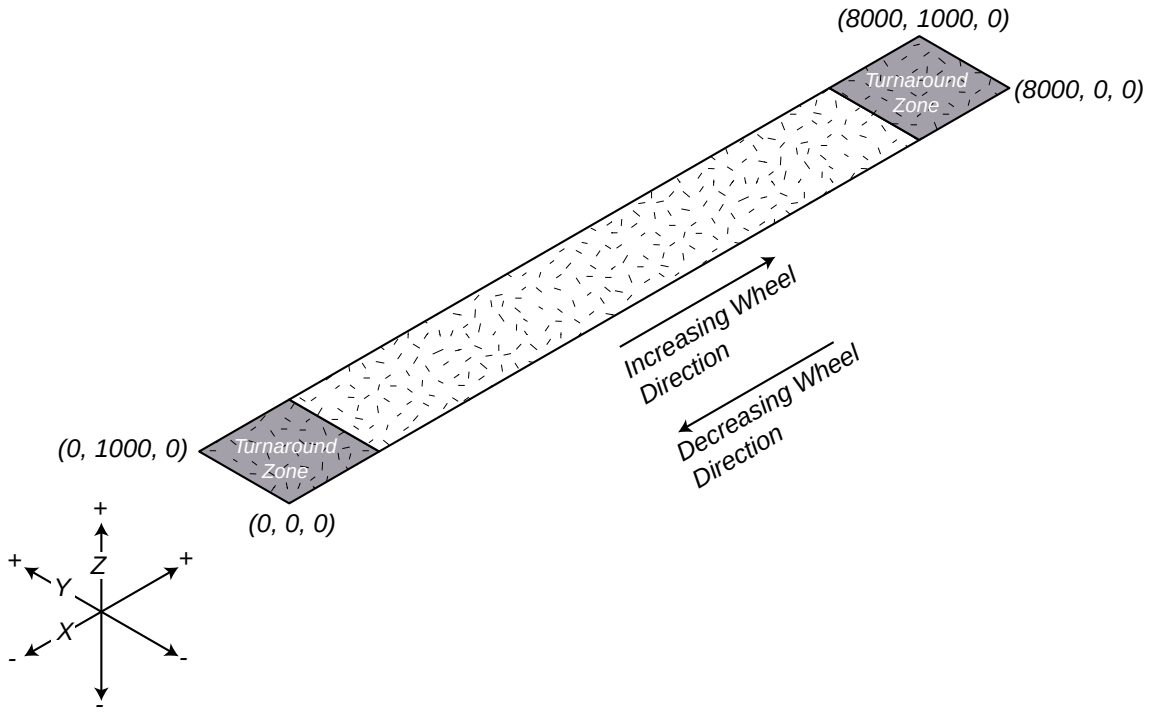


Figure 1. Orientation of HVS and test section.

In the PRC testing program, the layout for HVS tests has been slightly altered. First, a number of tests (for rutting on flexible pavements and all of the rigid pavement tests) were performed without wheel wander across the section, and so the real test section is less than one meter wide. In the rigid pavement sections, the instrumentation has been placed at the joints as opposed to at the numbered station locations along the test section. Because of these changes, it was decided that all location information in the database should be in terms of actual measurements on the section, and so the location of all instruments and the wheel are in terms of millimeters from the corner of the section which would be known as 0CS (i.e., the nearest left hand corner when viewing the section from the caravan side). The *x* axis runs along the section to the right and the *y* axis across the section away from the caravan side, with the *z* axis vertical. This method of instrument orientation is shown in Figure 2.

To maintain a normal coordinate system, the convention is positive upwards and negative downwards. Because of this, most deformations and deflections are negative. Although this is not normally how HVS data is presented, the charts of permanent deformation development, for example, seem to make more sense with this convention because they show the pavement deforming downwards. Instead of using left to right or CE-TE, which can be ambiguous, the database uses the term *increasing* (I) to refer to the wheel moving in a direction where *x* values are increasing and *decreasing* (D) for the other direction. It also uses *both* (B) if the section is being trafficked in both directions, and *environmental* (E) if the section is not being trafficked and is only being monitored for distress caused by environmental influences.

Isometric view of test section showing conventions for instrument location and orientation.



"Footprint" view showing orientation of test section beneath HVS with new coordinate system.

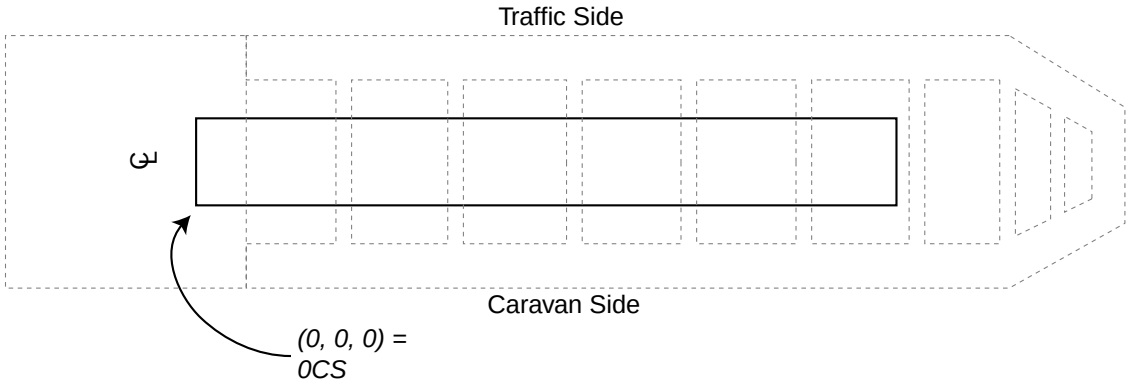


Figure 2. Test section coordinate system.

2.2 Problems with the old database structure

The old HVS database was structured around the data files produced by the old HVS data acquisition system. It included a single table to describe the section, containing one record for each section, and tables for each instrument, containing one record per data file. For each data file, a set of records was stored, one for each LVDT in the file (except the profilometer, which does not use LVDTs). Records were also inserted for each repeat, and for each line of raw data in the file, with the data for all of the LVDTs recorded in the same record. With this structure, there was no means of checking if data had been collected for the correct instruments, or if the header data for the various instruments matched. Also, this structure breaks many of the conventions for relational database design, making it difficult to construct queries on the database.

On the rigid pavement sections a long standing convention of having a single continuously increasing sequence of repeat counts over all of the loading for a section, was broken, and the repeat counter reset every time a new trafficking load was used. Each load was then treated as a separate test and given an additional letter after the HVS section number (e.g., 534aFD, 534bFD, 534cFD…). Because of this, an extra level had to be added into the database structure to accommodate these non-standard section codes, and to allow the reconstruction of a single repeat sequence for each section.

The database was developed in Microsoft Access, both because it is relatively simple to use and because it is easy to construct the queries needed. The loader was written using Microsoft Visual Basic for Applications as implemented in Microsoft Excel. It is a spreadsheet containing a number of pages with formulae and charts and several modules of macros. It was decided to implement the loader in Excel because the spreadsheet's internal functions could be used to manipulate the data during loading. Also, Visual Basic is fairly easy to understand, and

so the macros can be maintained by students with fairly minimal training and without the need for a compiler. The various macros are commented to aid in the maintenance of the code.

However, because the amount of data that has been collected is quite large, Access is not the ideal solution. In addition, a Web site has been developed to allow the data to be queried and viewed, and Access is also not suited for the types of concurrent queries that can occur on a Web site. Because of these issues, the database has been migrated to Oracle®, and the loader and web site have been designed so that they can operate with either Access or Oracle.

3.0 DATABASE STRUCTURE

The database structure is designed around that of the old database, but has been extended to handle the additional information from the various new instruments. The basic structure of the database is shown in Figure 3, and consists of section information, load history, instrumentation layout, header data and repeat data. Detailed printouts of the Access database structure and relationships are shown in Appendix A.

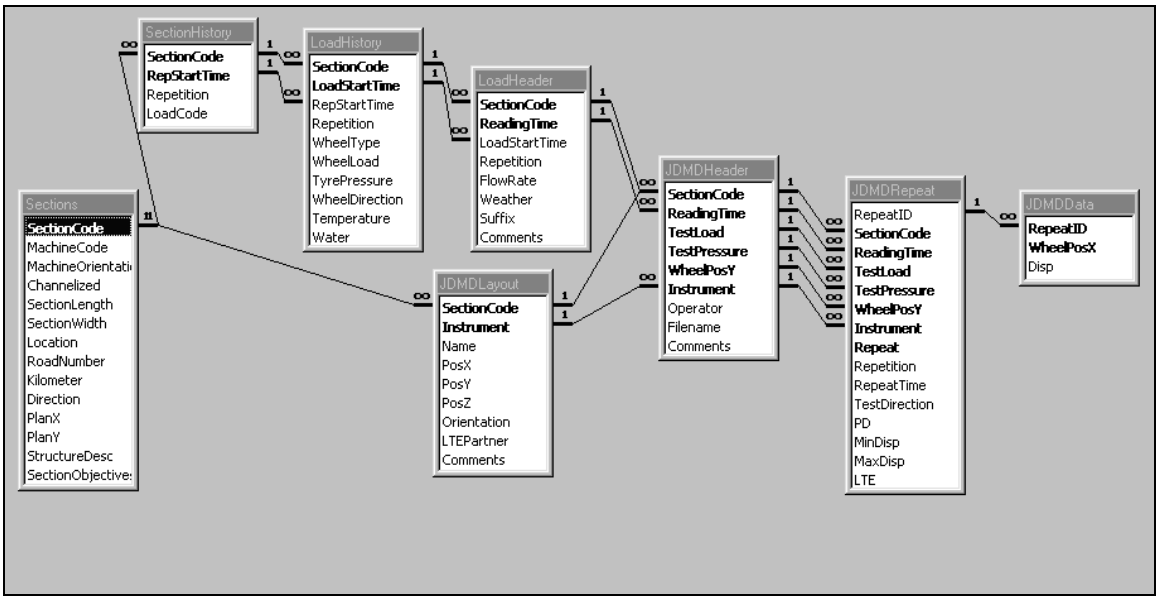


Figure 3. Core database structure example using a single instrument (JDMD).

Each of these various components are discussed in detail in the following sections, beginning with the section and load history information, and then dealing with the layout, header, and repeat information for each type of instrument. However, there is something to note in the table relationships shown in Figure 3, and that is that there are two paths of relationships leading to the load repetition information (which is the heart of the data collected from HVS test sections). The first is through a series of load history and header tables, which ensure that the load repetition data has the associated header information, such as the trafficking load being applied at the time. The second is though the instrumentation layout, which ensures that the load repetition data is attached to a valid instrument.

The PRC database, like the CSIR database, uses the reading time to index all of the data because multiple data sets can be recorded at any given repetition. However, it uses true date fields in the database, not the string representations used by the old database.

Within the database, all readings are in SI units. All lengths, deflections, or deformations are in millimeters (mm), all temperature data is in degrees Celsius (°C). Loads are in kilo-Newtons (kN), and pressures in kilo-Newtons per meter squared (or kilo-Pascals) (kPa). Strains are recorded in mirco-strains (με).

The data types for the various fields can be found from the database, but have been set to their minimum reasonable size. The raw data are stored as single precision floating point numbers, because this provides adequate accuracy while reducing the storage size of the tables.

3.1 Section and Load History

These tables are not instrument specific, and contain all of the information about the section itself, and the loading which has been applied to the section. There are four main tables: the `Sections`, `SectionHistory`, `LoadHistory`, and `LoadHeader` tables, and then three auxiliary

tables, which have not been altered from the old database: the Machine, Structure, and Material tables.

3.1.1 The Sections Table

The section information contains the section location and other details about a specific test section. Attached to this is the structure and materials data for the layers in the pavement, and information about the HVS used to test the section. Because this additional information has not been altered from the old database, it is not dealt with in detail here.

The Section table (Figure 4) has a single field in its primary index, which is the SectionCode, and this is used to index all of the data for that section. The additional fields in the Sections table are: the MachineCode, which is linked to the Machine table and stores which HVS was used to test the section, and MachineOrientation, which stores whether the increasing section direction is either from cabin to tow end or tow to cabin on the HVS, and Channelized, which indicates whether the loading beam was wandering or not.

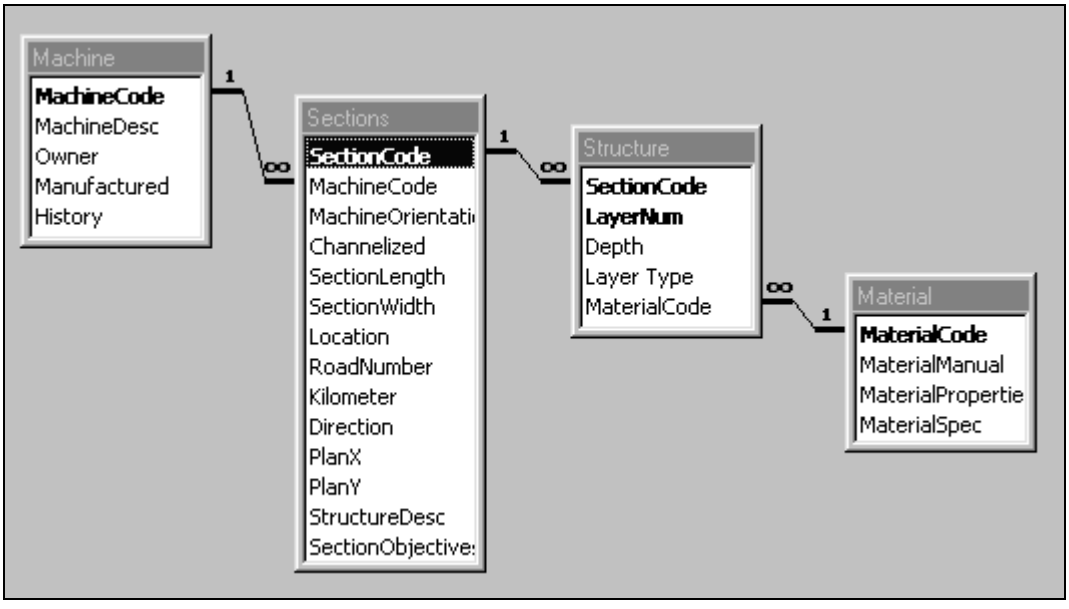


Figure 4. The Sections Table.

The SectionLength and SectionWidth are stored, along with location information, in terms of a Location field, which is a text string containing the colloquial site name, the RoadNumber, Kilometer and Direction if the section is on an in service pavement, or the PlanX and PlanY coordinates, if the section is on a specially constructed test pavement. There are also two more text fields, StructureDesc for a structure description, and SectionObjectives for a brief overview of the section objectives.

The Machine table contains some minimal information on the specific HVS used to test the section, while the Structure table contains records for each layer in the pavement, providing some basic information on the structure of the pavement. This table is linked to the Material table, which stores some basic information on the material used in each layer. In future work, it is planned to extend these last two tables to provide full information about the section, including links to laboratory testing results.

3.1.2 The SectionHistory Table

Shown in Figure 5, this simple table handles the resetting of the repetition counter on many of the rigid pavement sections, as discussed in Section 2.2. The new section code (such as 534aFD) is known as a LoadCode and is indexed to the SectionCode by a repetition start time, which is the time at which the loading was changed. The real repetition at the time is also recorded, but this is calculated automatically by the database through analysis of the LoadHistory table. The value of the repetition start time is also automatically computed by the database.

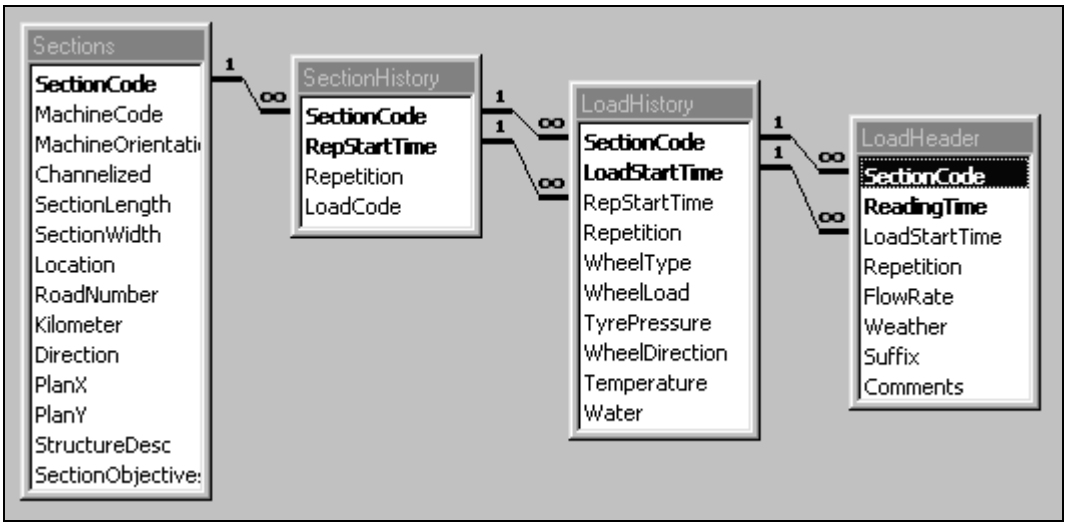


Figure 5. The `SectionHistory` and `LoadHistory` Tables.

When the loader encounters a new data file, what is recorded as the section code in the data file is checked against the list of load codes to discover the real section code. Whenever queries that require the repetition count to be displayed are performed on the database, the value of `Repetition` from this table is added to the repetition values stored in the other tables to produce a single continuously increasing sequence of repetitions for the section.

Because of this resetting of the repetition counter, it is a requirement that the section be given a new load code if the repetition counter is reset, and that the repetition counter be reset if a new load code is given to the various phases of loading.

In an idealized APT database, this table and the concept of load codes would not be needed because the repetition counter should never be reset.

3.1.3 The `LoadHistory` Table

The `LoadHistory` table, also shown in Figure 5, stores all of the information related to the trafficking of the pavement. It is indexed under the `SectionCode` and `RepStartTime` by a load start time, which is the time at which that loading phase was begun. Both the `LoadStartTime` and

Repetition are calculated automatically by the database, although the repetition can be manually edited and will not be corrected if its value is reasonable. The table records the WheelLoad, the TyrePressure, and WheelDirection, which can be either "I," "D," or "B" (for increasing, decreasing or both). The WheelConfiguration used on the machine, which is limited to "Single," "Dual," "Super," or "Aircraft" (although additional values can be added), is also stored in this table because on some sections the wheel was changed part way through a test.

Additional information in this table is the nominal section temperature (based on the test plan, not on the actual temperature measurements), which is left empty if there is no temperature control, and whether or not water was being added artificially to the section.

3.1.4 The LoadHeader Table

Although the LoadHeader table (Figure 5) conceptually forms part of the header information, it is convenient to discuss it here, because it is related to the loading of the section and not to any instrument in particular. This table contains one record for every time data is recorded for the section, no matter which instruments are read. The table is indexed by the SectionCode and ReadingTime, which is the time at which readings are begun. This time is automatically calculated by the database during the loading of new data files. The Repetition stored in this table is the value from the data file—it is only adjusted later during querying to compensate for resets of the counter. Also in this table are a water flow rate (FlowRate, for artificially added water), a text description of the weather (Weather) and a text comment (Comments).

There is an additional field known as Suffix, which is only used on data files recorded using the old DAS. With the old system two letters, beginning at AA, were appended to the end

of the DOS filename used for the data files. This suffix can be used to match data recorded at the same time, which should have the same header information (specifically the repetition count and wheel load). With the new system, data is recorded at the same time, and so all of the data files have the same `ReadingTime`, and there is no need for a suffix.

3.2 Joint Deflection Measuring Device (JDMD)

JDMD's are single linear voltage displacement transducers (LVDTs) which are either mounted vertically at the longitudinal center of a slab along the slab edge, vertically in pairs spanning a joint along the longitudinal edge, or horizontally across a joint to measure the horizontal joint movement. They were developed for the testing of rigid pavements by the PRC, and are the primary instruments for measuring the response of rigid pavements. Although the old DAS was modified to read JDMDs, they had to be read either in pairs or singly, since the 256 points measured by the old system did not span sufficient length to capture the full response of the instrument as the wheel moved over the slab (rigid pavement HVS sections are normally centered on a slab, with two joints in the test section, each about one meter from the ends of section). The new DAS measures response over a longer length and all of the instruments are read at the same time, so there is only one data file for all of the JDMDs on a section.

The structure of the JDMD data tables is shown in Figure 6, where the layout, header and repeat components can be seen.

3.2.1 The `JDMDLayout` Table

The base table for any instrument is the layout table (in this case, `JDMDLayout`). This table contains the information about the location of JDMDs on the section, with one record for each instrument. Each instrument is assigned an `Instrument` number, which is internal to the

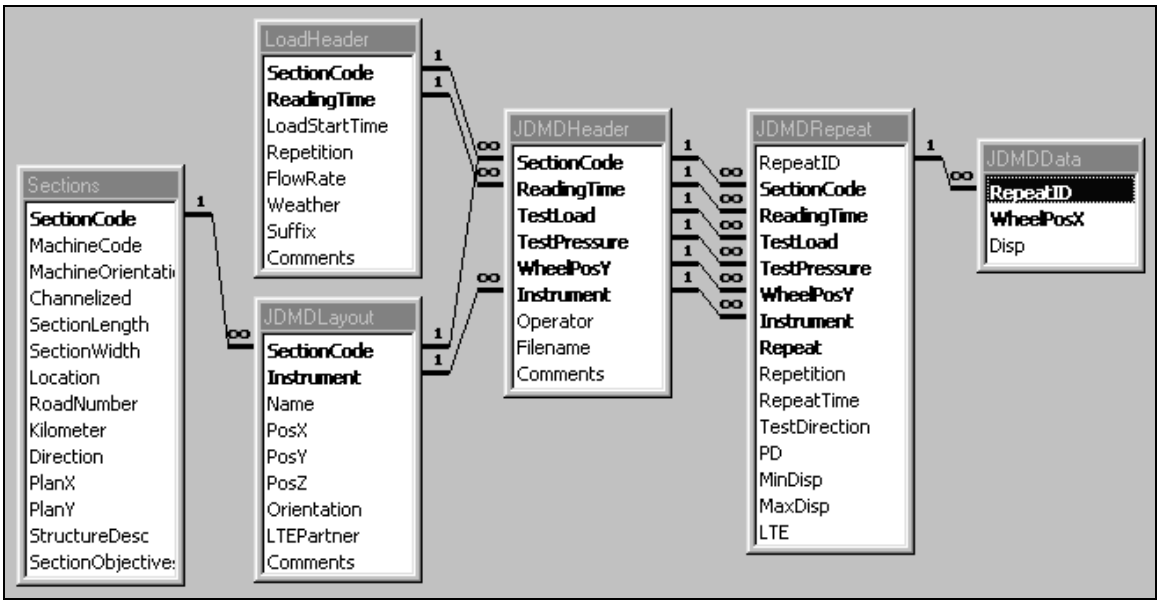


Figure 6. JDMD data structure

database. Each instrument also has a `Name`, by which it is known in the raw data files and in all of the outputs, which must be unique within each section. The position of each instrument is recorded in terms of its *x*, *y* and *z* coordinates (see Section 2.1 for details on the coordinate system used by the database) and an orientation, which is recorded as the axis along which the instrument is aligned (for example, for the vertically oriented JDMDs, this is "Z"). By convention, instruments can also be specified by an axis of rotation and an angle – the instrument is assumed to be in the plane of the other two axes (so "Z:45" implies that the instrument is in the *x*-*y* plane, at an angle of 45° to the *x*-axis. A negative sign in front of the axis specifies that, by default, the instrument readings need to be inverted to place them in the correct sign convention for the database. If the `Orientation` is "0" (zero), then the data for that instrument is ignored during loading.

Because JDMDs can be set up as pairs across a joint, to measure the Load Transfer Efficiency (LTE) of the joint, the table also contains an `LTEPartner` field, which is the JDMD

number of the instruments partner. This is not an optimal arrangement for checking if the instrumentation layout is correct, but the validity of these partner relationships (which must be reciprocal) is checked while the data is loaded. The `Comment` field is a text field for comments.

3.2.2 The `JDMDHeader` Table

Each time readings are observed for JDMDs, a record is added to the `JDMDHeader` table. Each header record must have a corresponding `LoadHeader` record, which stores the general details (like the repetition), but there can be multiple `JDMDHeader` records per `LoadHeader` record, which are differentiated by the test wheel load (`TestLoad`), test tire pressure (`TestPressure`), the lateral position of the wheel (`WheelPosY`) and the `Instrument` of the JDMD being recorded. The `Filename` from which the data was obtained is recorded, along with the machine `Operator` and comments. Because the JDMD number appears in the header table, the same filename can appear multiple times in the `JDMDHeader` table. This is needed because the old DAS recorded data in multiple data files, while the new DAS records in one data file.

The database relationships ensure that a `LoadHeader` record and a `JMDMDLayout` record occur for each `JDMDHeader` record, so that data cannot be collected when the repetition is not known, or for instruments for which positions are not known.

If a new APT database were being developed from scratch, this table should only contain the section code and reading time as primary keys, and the filename should be unique: implying that all data for a set of JDMD readings (no matter what test load) is stored in a single data file. This table should not be linked to the layout table.

3.2.3 The JDMDRepeat and JDMDData Tables

For each header, there can be a number of repeats, with each repeat representing a full deflection bowl for the instrument. The JDMDRepeat table includes all of the primary key fields from the JDMDHeader table and adds only a Repeat number to the key. The table also stores the actual repetition count when the data was recorded (which might differ slightly from that of the LoadHeader record), and the actual time when the repeat was performed. The direction of wheel movement is stored in this table, and in most queries the direction of movement of the wheel is ignored, since its effect is assumed to be negligible.

The JDMDRepeat record also stores level one summary information for the deflection bowl, including the permanent deformation (PD), the peak downwards deflection from the PD (MaxDisp), the peak upwards deflection (MinDisp) and the LTE for the instrument (if it has a LTE partner). Section 4.2.4 describes how this information is obtained.

Attached to the repeat table are JDMDData records, which contain the raw data points for the deflection bowl for that repeat, with one data point per record. Because there is a large number of points, and because the repeat table has a large number of fields in its primary index, a special RepeatID (which is an automatically generated and unique 32-bit integer number) is used to index this data.

The database is specifically designed to permit operation with or without all of the deflection bowl data loaded (the JDMDData records), so that the size of the database can be reduced if desired.

3.3 Multi-Depth Deflectometer (MDD)

MDDs consist of a number of LVDTs that measure the deflection of various layers at depth in the pavement relative to an anchor point 3 m beneath the pavement surface. The data structure for the MDDs (Figure 7) differs from that of the JDMDs in only one detail: because of the nature of MDDs, they have several modules stacked in one hole. Thus the layout table is split into a table which contains the *x*,*y* coordinates of the hole, and a second table which contains the *z* coordinates of the individual modules. This means that modules can only occur within a defined hole.

3.3.1 The `MDDLayout` and `MDDModules` Tables

As mentioned in the previous section, the MDD layout information is split between two tables: the `MDDLayout` table containing information on the MDD hole, similar to the JDMD layout, and the `MDDModules` table such as its position.

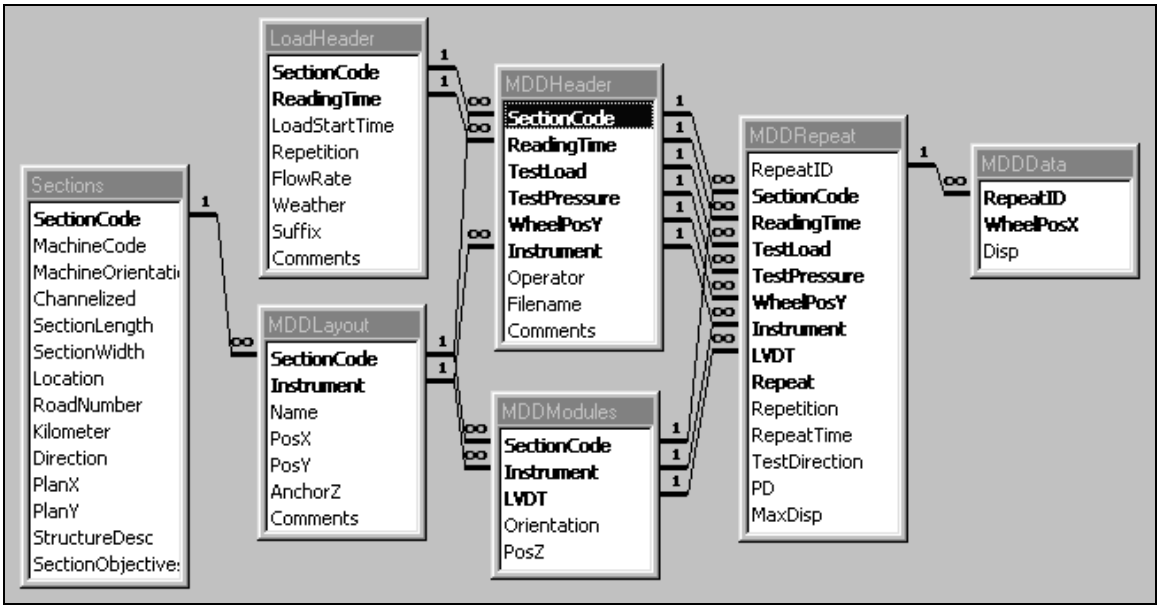


Figure 7. MDD data structure

MDDs are also assigned an `Instrument` number, but are normally indexed through their `Name` (traditionally the MDD is named after its longitudinal position, so MDD8 would be four meters from the end of the section, at the centerline).

The `MDDModules` table only stores the module's `LVDT` number (which is defined by its position in the stack), and the depth (which will always be a negative number). There is an orientation field, but it is limited to "Z" or "–Z" because the modules are always along the *z* axis.

3.3.2 The `MDDHeader` Table

The `MDDHeader` table does not differ from the `JDMDHeader` table, except that under the data collection procedures of both the old and new DASs, there is only one data file per MDD, and so the filenames in this table are unique.

3.3.3 The `MDDRepeat` and `MDDData` tables

These tables are almost identical to the corresponding JDMD tables, except that there is no LTE or upwards displacement in the repeat table. The MDD has no concept like load transfer efficiency, and normally any upwards deflections are not of interest.

3.4 Road Surface Deflectometer (RSD)

The RSD is a modified Benkelman beam, which measures the deflection of a point on the road surface as the loading wheel passes over the point. Normally a number of points are stored in the same file, although with the old DAS the data for points along the centerline are normally stored in one file, and the off center points in another.

On some sections the RSD has been used to determine if there is movement of the MDD or JDMD anchors, but this data is not stored in the database, because it is not pavement response data.

The data structure for RSD data is shown in Figure 8.

3.4.1 The RSDLayout Table

This table is similar to the JDMDLayout table, except that there is no LTE partner. The *z* coordinate is almost always zero because the RSD measures the deflection at the surface. The only time this might change is if the RSD were used to measure deflections on some type of rod anchored at depth in the pavement. The orientation is limited to "Z" or "–Z."

3.4.2 The RSDHeader Table

The RSDHeader table has a structure similar to the MDDHeader table, except that there is no connection to the layout table. This is because the RSD readings for one lateral wheel position

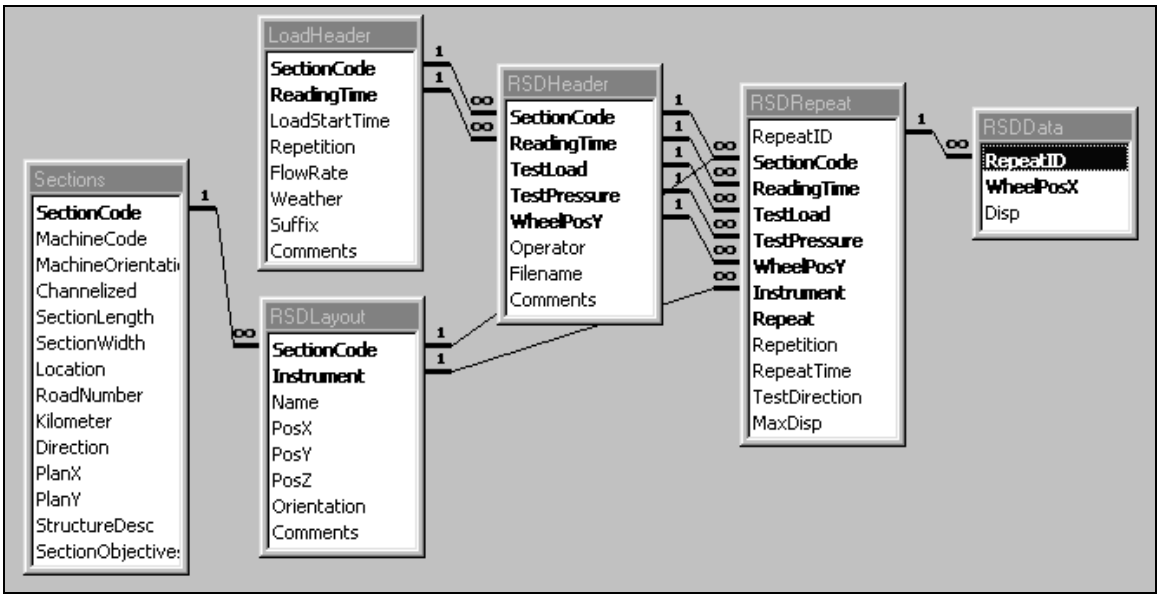


Figure 8. RSD data structure.

are always stored in the same file. This is sufficient information to characterize the header. If at any time the results for individual RSD points are stored in separate files, then the RSD number would have to be added to this table.

In a new APT database all of the RSD data, all loads should be stored in a single file and everything but the section code and reading time should be excluded from the primary key.

3.4.3 The RSDRepeat and RSDData Tables

These tables are very similar to the corresponding JDMD or MDD tables, except that there is only a peak downwards deflection, since the RSD cannot be used to determine permanent deformation.

3.5 Crack Activity Meter (CAM)

The CAM consists of two small LVDTs, one that measures vertical movement of a crack, and the other the horizontal movement. The device is glued across a crack and so measures relative movement of the two sides of the crack. The data structure for CAM data is shown in Figure 9.

3.5.1 The CAMLayout and CAMModules Table

The CAMLayout table is similar in design and purpose to the MDDLayout table, except that the *z* coordinate of the CAM is also stored. This is because it is theoretically possible to mount the CAM below the road surface (for example, on a crack in a rigid pavement that has been overlaid and then cored and the CAM placed into the core hole).

The CAMModules table is like the MDDModules table, except that it stores the orientation of

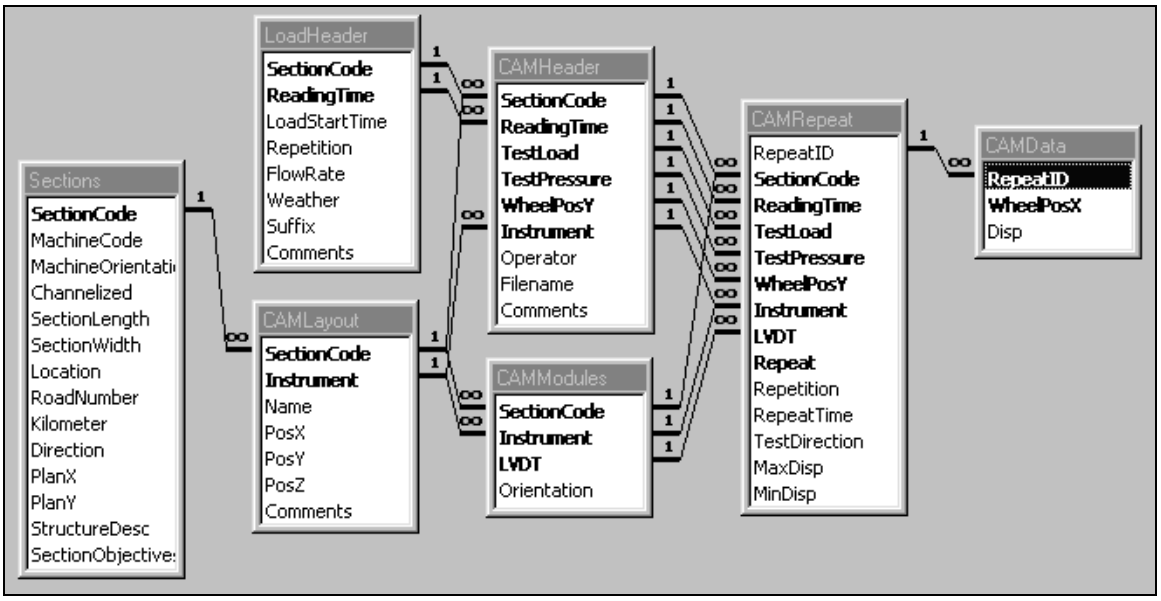


Figure 9. CAM data structure.

the LVDTs rather than the *z* coordinate. The orientations are normally 'X' and 'Z' although they depend on the orientation of the crack on the section.

3.5.2 The `CAMHeader` Table

The `CAMHeader` table is identical in design and function to the `MDDHeader` table. Normally the results from each CAM location are stored in separate data files, so the data file names are unique, but this is not enforced within the database structure.

3.5.3 The `CAMRepeat` and `CAMData` Tables

These tables are almost identical in design and function to the corresponding MDD tables, except that for the CAM a maximum and minimum deflection are stored, and no permanent deformation is stored. This is because the opening and closing of the crack, which can both occur as the load passes, must be captured.

3.6 Strain Gauges

Strain gauges have primarily been used on rigid pavements, where they have been placed during construction. However, it is conceivable that they will be used on flexible pavements, and that they might be retrofitted during testing.

Two kinds of gauges have been used: single gauges, and 45° rosettes. With both DAS's all of the strain gauges are read simultaneously and so there is only one strain gauge data file per set of readings. The data structure for strain gauges is shown in Figure 10.

3.6.1 The `StrainLayout` Table

All of the strain gauges are stored in a single table, which stores their *x*, *y* and *z* positions, along with their orientations. This table could have been split into a layout and modules table, with the orientation of the gauges in a rosette stored in the modules table, but this complicates some of the queries, and so has not been implemented. The queries are complicated by the fact

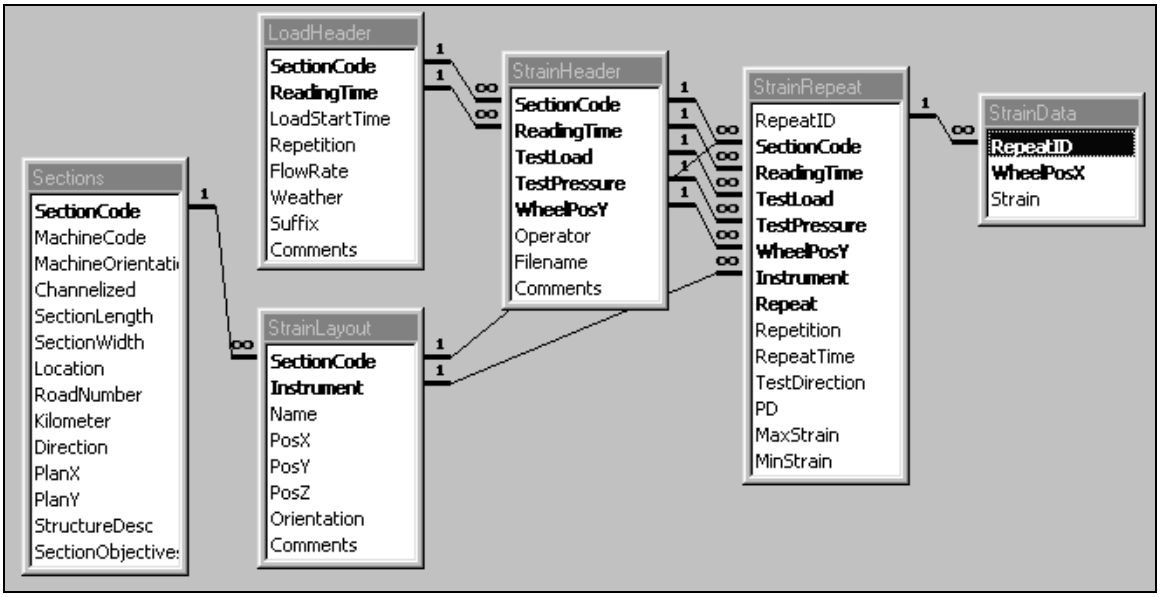


Figure 10. Strain gauge data structure

that the new DAS names each strain gauge with a unique name, and so the name would have to be moved down into a `StrainModules` table if it were created. However, all of the rest of the tables contain a unique name in the layout table, and so special queries would be needed for the strain gauges. The DAS software should be modified to have groups of strain gauges, with one name, if the addition of a strain modules table becomes necessary in the future. As with all of the other layout tables, the gauges are assigned a number by the database, but are known by their unique name.

3.6.2 The `StrainHeader` Table

This table is similar in design and function to all of the other header tables. Because the strain gauges are recorded simultaneously, the filename is unique.

3.6.3 The `StrainRepeat` and `StrainData` Tables

These tables are similar in design and function to the corresponding tables for the other instruments, except that the displacements are now strains (stored in terms of micro-strain, with positive strains representing compression) and the maximum and minimum strain is stored (since the gauge could be in both tension and compression).

3.7 Laser Profilometer

The profilometer measures the transverse profile of the pavement surface to determine rutting and is thus only really meaningful on flexible pavements. The profilometer has been tried for measuring concrete slab curvature, but it is not long enough to provide reliable data for this purpose.

The operation of the profilometer is significantly different from the other instruments. The other instruments all measure the dynamic response of the pavement to loading from the HVS wheel while the instrument is fixed. In contrast, the profilometer does not involve the HVS wheel at all. Instead, the profilometer utilizes a beam that is placed transversely across the test section. A laser range finder moves along this beam and measures the distance from the beam to the pavement to generate a profile of the pavement at the given location. This is typically repeated at all numbered stations on the test section (0 to 16). The original profile must be compared with the new profile to determine the degree of rutting between measurements (this is further described in Section 4.3.3). The data structure for profilometer data is shown in Figure 11.

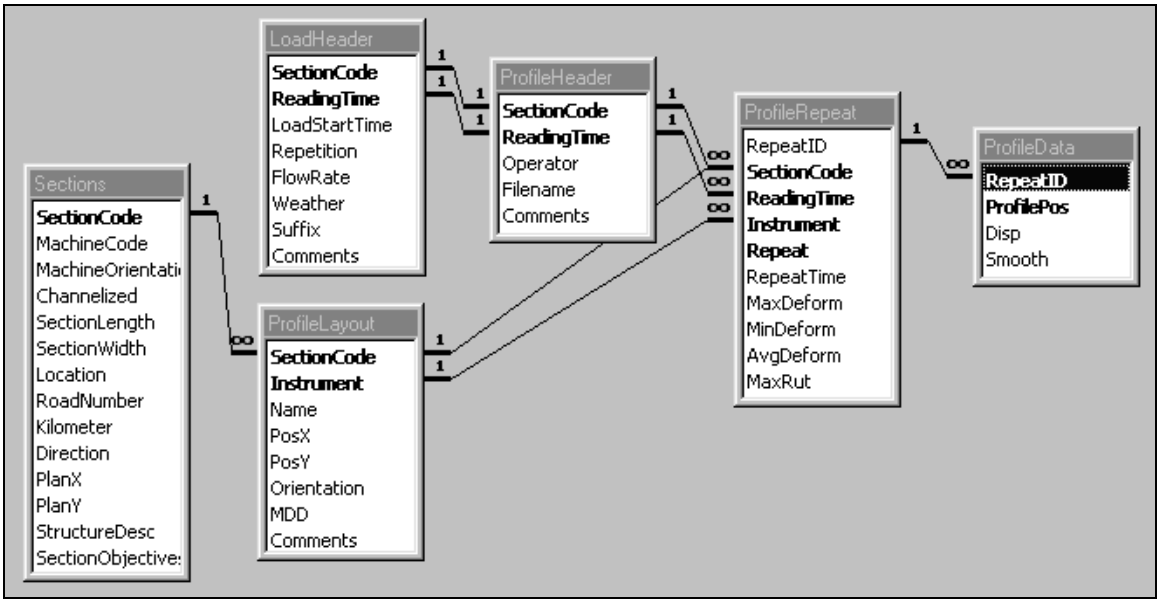


Figure 11. Profilometer data structure.

One drawback of the processing required for the profilometer, is that the level one processing cannot be performed within the database queries, and so processed results for the profile have to be stored along with the original data.

3.7.1 The ProfileLayout Table

While this table has similar structure to the other layout tables, it is used in a different fashion. Firstly, the *x* and *y* coordinates are for the first reading from the laser (which is actually fairly difficult to measure). Secondly, the orientation describes the direction along which the profilometer beam is pointed (the readings from the profilometer are always in the *z* direction). An additional field, MDD, can be used to indicate that there is an MDD top cap along the profile, which might influence the rut depth calculation.

3.7.2 The ProfileHeader Table

Because measurement with the profilometer does not involve the loading wheel, this table does not have any load information, and so has a one-to-one relationship with the LoadHeader table. This is only feasible because all of the profiles at a particular time are stored in one data file. If they were split into separate files then the profile number would have to be added to this table.

3.7.3 The ProfileRepeat and ProfileData Tables

Although these tables are similar in structure to the other instruments, they are also missing the load information and they contain slightly different information. The MaxDeform, MinDeform, AvgDeform, and MaxRut fields store the maximum (negative) deformation from an original surface, the maximum (positive) deformation, the average deformation, and the maximum total rut depth. These are explained in Section 4.3.3.

The data table contains two data fields: the raw displacement measured by the profilometer (`Disp`), and the smoothed processed profile (`Smooth`), as a deviation from the original profile.

3.8 Thermocouples

Thermocouples are also significantly different from the other instrumentation. They are recorded at the same time as the other instruments with the new DAS (with the old DAS they were recorded at hourly intervals throughout the day). Because of this data collection schedule, the thermocouple data is not linked to the other data directly. This can be seen in Figure 12, in which there are thin lines showing only a casual relationship between the `ThermoHeader` and `LoadHeader` tables.

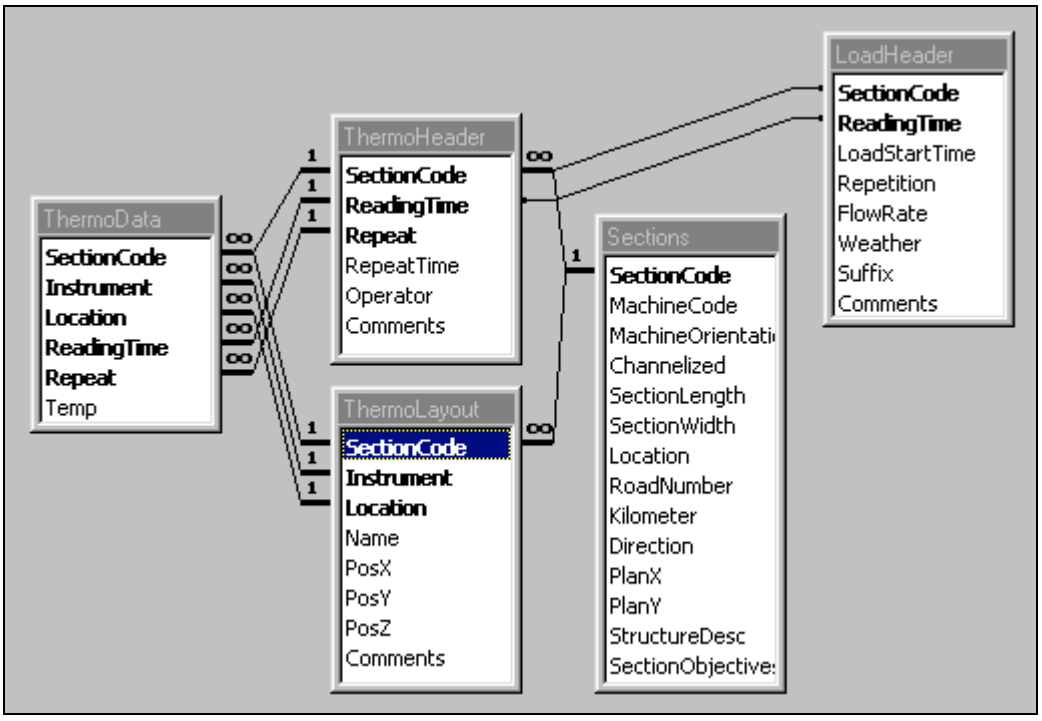


Figure 12. Thermocouple data structure.

3.8.1 The ThermoLayout Table

There are three kinds of locations for thermocouples: Outside, Climate chamber, and Pavement, with *x*, *y*, and *z* information only for the pavement gauges (since the other gauges do not have positions within the section). Once again, each gauge is assigned a number (which is unique within the various types of locations), and each gauge has a unique name by which it is known.

3.8.2 The ThermoHeader Table

The thermocouple header table stores the time at which a set of readings was taken, along with the repeat number if more than one repeat was taken, and the time at which any additional repeats were taken. These fields would normally be stored in a repeat table, but because there is less thermocouple data it was decided to merge these two tables to simplify queries on the database.

3.8.3 The ThermoData Table

The ThermoData table contains one record for each gauge measurement, with the temperature stored in degrees Celsius. The tables are related in such a way that data can only be stored for a valid gauge (i.e. one which exists in the layout table), and at a time for which there is a header record.

4.0 DATA LOADING AND PROCESSING

A large amount of pavement response data, particularly dynamic deflection data is being captured from HVS test sections. In many cases this data is being captured at rates which are higher than the rate at which it can be processed by hand to extract meaningful parameters about

the pavement's behavior, and so automated data processing is needed. In particular, the data must be verified, and accurate and precise first-level analysis parameters, such as maximum dynamic deflection and accumulated permanent deformation, extracted. Within the PRC-HVS database, these functions are performed within the loader, which is a Microsoft Excel spreadsheet.

This section details the techniques used for the automatic verification, cleaning, and analysis of pavement response data, and the subsequent loading of that data into the database. However, it is convenient to first address the algorithms used to load the data. After that, the issue of data verification, to determine if the data presented is in fact a valid set of pavement response data, is addressed. Finally, the cleaning and filtering of the data, followed by the extraction of first level pavement response variables, is covered.

4.1 Data Loading Procedure

The PRC-HVS database loader is a spreadsheet, `PRC-HVSLoader.xls`, which is simply opened in Microsoft Excel. On the first sheet, three cells contain the setup information for the database:

- The first cell is the directory where new data files can be found, for example, a directory where they are saved from email attachments, or the computer's floppy disk drive.
- The second cell is the directory where the raw data files are stored. When new data has been successfully loaded into the database, the files are moved from the first directory to a child directory under the second directory. This directory structure has

two levels: the first is the section code, the second the instrument type. This directory might be a directory on the computer's hard disk, on a network server, or a CD-ROM.

- The third directory is the directory which contains the Microsoft Access database, `PRC-HVS.mdb`. If the database is supplied on a CD-ROM, then it should be copied onto the computer's hard disk, since the database was not designed to be operated with a read-only database file.

4.1.1 Loading New Data

The procedure for loading new data is very simple, provided that the data files do not contain errors. They are simply copied into the directory for new data, and a button that appears on the first sheet labeled, "Load New Data," is pressed. Visual Basic macros then perform all of the processing and loading of the data.

The loading procedures for each instrument are very similar, and so only the JDMD is dealt with as an example in this section in order to outline the procedure. The comments within the macro code for the various procedures provide a better set of documentation for the actual logic and algorithms that are applied.

Pressing the "Load New Data" button calls a macro named `Load_New_Data`, which builds a list of all of the files in the incoming data directory. It then tries to establish, based on the filename, the instrument type associated with the data contained in each file. For example, JDMD data files from the old DAS have the extension .JAM and for the new DAS they contain "-JDMD-" and have the extension .CSV.

If a JDMD data file is found, then `Load_New_JDMD_Data` is called with the file name. The `JDMDHeader` table is queried to determine if the file has already been loaded by the database. If it

has not, then either `Load_JDMD_Data_CSIR` or `Load_JDMD_Data_PRC` is called depending on whether the data is from the old or new DAS.

The procedures for the old and new systems follow a similar pattern for loading the data, except for differences dictated by the different file formats. First the raw data file is opened, and the header information from the file is read. This header information is used to determine the correct section code, based on the load code (described in Section 3.1.2). If the section or load code is new, then a dialog box appears that allows the operator to input the correct information for the section. The instrument name and location are also extracted from the data file and `JDMDCalcLayout` is used to add the instruments (also via a dialog box) to the database if they are not yet known. While processing this header, the correct `LoadStartTime`, `ReadingTime`, and `Instrument` numbers are also determined so that they can be used later when loading the data. The header information is then written to the spreadsheet page (`JDMDCalc`) on which the calculations for the JDMD are performed.

The loading procedure then loops over the data file loading the deflection bowl data for each repeat. All of the calculations are performed by one function (`JDMDCalcData`) which is called by both procedures to ensure that the calculations are performed in the same fashion. Once the calculations are performed, another function (`JDMDCalcLoad`) is called which loads the data in the database.

The loader for the old DAS has to perform some additional steps to correct the wheel longitudinal position since the old DAS does not provide the actual wheel position. Firstly, the direction the wheel was moving is not known, so a direction is assumed based on the position of the instrument (i.e., that the wheel started at the other end of the section from the instrument). The reading number (1-256) is then multiplied by the distance between readings (which is in the

file header) to obtain the wheel position in millimeters. However, the wheel never starts at the end of the section, and so a second adjustment based on the location of the peak deflection needs to be performed to correct the wheel position. The procedure used for this is explained in Section 4.3.1.

Because the old DAS required the header data to be manually entered during data collection, the files contain a fairly large number of errors, and so, as the header data is being loaded, it is checked to ensure that the file's data is at least consistent with the other data files which have been loaded. After the correct section code has been established, the file's suffix (the two letters appended to the DOS filename) are used to compare the file to other files which have been loaded. If files with the same suffix have been loaded, then the repetition and trafficking loads are compared to ensure that the data matches.

If the data file appears to contain an error in the header data (for either system) then the loader will display an error message detailing why it could not load the file, and the file is left in the new data directory. If the loader's algorithms detect that one or more of the deflection bowls in the data files is suspect, then that deflection bowl is ignored, but the file is still loaded and copied into the backup data directory.

After the data has been loaded, a number of procedures are run to check the validity of the data which has been inserted into the `SectionHistory`, `LoadHistory`, and `LoadHeader` tables, and to recalculate the start times and repetitions if necessary.

4.1.2 Data Management

The loader also contains some facilities to perform some basic data management, although more complex tasks still need to be performed directly in the database. These tasks fall

into two main categories: editing of section and instrumentation information, and loading and unloading deflection bowl data.

The first few sheets of the loader spreadsheet contain tables that display the section and the instrumentation and load history information for the section in question. If the section number is edited, then these tables are automatically updated from the database. These tables can be edited in the spreadsheet if they contain errors. The spreadsheet also contains buttons that cause the macros to update the database from the edited information.

The `ManageData` sheet in the loader also contains a number of buttons that can be used to unload the raw deflection bowl data for various instruments, or to reload the information from the raw data files. Because the database was designed to operate without the raw data, this does not affect the level one data stored in the tables—the only effect is on whether the raw deflection bowls can be viewed. On this sheet are two cells that control the first and last repetition, which is used for the unloading or reloading.

Also on this page are a series of radio buttons to control whether the database uses filtered (smoothed) deflection/deformations for the calculation of the level one information, and whether the database loads the raw deflection bowls while loading new data.

4.2 Data Verification

Throughout this section, it is assumed that the data being presented is an ordered set of n data pairs:

- **an *x* value,** which represents a distance (either the location of the load or the location of the instrument); and
- **a *y* value,** which is the recorded value of the instrument in whatever units of measure are appropriate.

In matrix notation, this can be represented as [**x**,**y**] assuming that **x** and **y** are column vectors.

It is also assumed that other verification (such as matching the data to the correct instrument) has taken place at a higher level within the data processing system. The data verification procedures operate entirely independently of the type of data that they are processing.

The first thing that needs to be established is whether the data set is a valid set of readings for the instrument. Typical problems might be that the instrument was not actually being read—that the data is just channel noise, or that the instrument was out of range, or went out of range while recording.

The first check of the data is to determine if **x** is continuously increasing or decreasing. To do this, the locations of the minimum and maximum reading position (x) in the set need to be determined. The minimum and maximum readings would ideally be the first and last values respectively, but in practice there may be some noise (like repositioning of the wheel) at either the beginning or end of the set. If the number of data points between the minimum and the maximum is less than 90 percent of the full data set of n points, then the data set is rejected, because then more than 10 percent of the data consists of segments for which there were inconsistencies in the position. If the minimum and maximum are not the first and last readings, then inconsistent data from the start or end is discarded. After this, the **x** data is checked to determine if it is monotonically increasing (i.e., $x_{i+1}>x_i$) or decreasing (i.e., $x_{i+1}<x_i$), and if this is not the case, the data set is also rejected.

Instruments for which several sensors are recorded simultaneously require that their positions be checked only once, and so they are handled by a separate procedure. Once the

location (**x**) data has been validated, the measured data can be checked. The first check that needs to be performed is to determine if the measured data actually reflects real data, or if it is just random noise due to a fault, such as a loose connection or a misconfigured channel. This is often very difficult to determine because the real data can often have the same appearance. These checks should be performed by the machine operators when the data is collected. Typically some invalid data makes it through to the database loader.

There are two distinct data cases to analyze: static (or environmental) response and dynamic response. In the static response mode, it is assumed that y_i is actually a repeated measurement of a single value y_0, and that x_i is a dummy variable. In the dynamic response mode, it is assumed that y_i represents some change with a changing load position or instrument location.

In order for the data verification procedures to be instrument independent, a parameter ε, which is the precision of measurement of y, is defined for each instrument. For deflection data, ε is typically 0.005 mm; for strain data, ε is typically 10 με; and for profile data, ε is typically 0.05 mm. Obviously, if **y** contains any non-numeric quantities the data set is rejected.

One common problem is that the instrument may go out of range during testing. If this occurs, it needs to be detected and the data set rejected. If no errors have been found in the data set, then the next step is data filtering, which has two functions: to clean spurious measurement errors from the data and to smooth unrealistically noisy data. The algorithms used in these two operations vary considerably. During the process of removing spurious data points, as little data as possible is touched and the measurements from spurious data points are completely removed. To remove noice, all of the data is processed.

4.2.1 Out of range data.

Both static and dynamic data need to be tested to determine whether the channel contains no data, or if the gauge was out of range. The first test is simple: if the difference between the maximum and minimum measurement is less then one tenth of the instrument's precision (i.e., $\max(y_i)-\max(y_i)<\varepsilon/10$), then the data is rejected as containing no information, since even for static data measurements there is some variation in the recorded data values.

The second test is a little more complex because it requires looking for flat maximum or minimum data groups. This is done by looping over the data and counting the number of consecutive points that are equal to the maximum or minimum. If this number exceeds a given threshold, then the data set is rejected.

The allowable threshold can be tuned, but since instruments are likely to go out of range at the critical point of interest (the peak of the deflection bowl), it should not be too large, else data that misrepresents pavement behavior will be accepted as valid. After some experimentation, the threshold for dynamic data was set at 5 consecutive points. For static data, the threshold was set at 20 points since there might be long sets of constant data and out of range data is not as likely on static measurements.

4.2.2 Data cleaning

The algorithm used for data cleaning has been specifically tuned for pavement response data. It makes use of the fact that this data changes relatively slowly compared to electronic noise. The algorithm is completely independent of the instruments and depends only on the precision ε and whether the data is static or dynamic. In addition, the algorithm makes use of

two of parameters derived from the data: the range between the maximum and minimum values and the standard deviation (σ) of the data from its nine point moving average (see Section 4.2.3).

First, the deviation of each data point from an 8 point moving average is calculated as:

$$d_i = \left| \left(\sum_{j=i-4}^{i-1} y_j + \sum_{j=i-1}^{i+4} y_j \right) \Big/ 8 - y_i \right| \quad \textbf{(1)}$$

This is then compared with a threshold value to determine if the deviation of the point is too large. The data threshold is calculated as the minimum of 5 percent of the measured range of **y** for most data points, or 3σ for dynamic data or 5σ for static data. If the dynamic measurement being tested is a minimum or maximum value, then only the 3σ criteria is applied because these values are critical to later calculations, while small anomalies within the bowl are not as critical. Mathematically this can be represented (for dynamic bowls) as:

$$d_{\max} = \begin{cases} \max\left(2\varepsilon, 3\sigma\right) \text{ if } \left\{y_i = \max\left(y_i\right)\right\} \cup \left\{y_i = \min\left(y_i\right)\right\} \\ \max\left(2\varepsilon, 3\sigma, 0.05\left(\max\left(y_i\right) - \min\left(y_i\right)\right)\right) \text{ otherwise} \end{cases} \quad \textbf{(2)}$$

Two additional checks are performed on each point for which the deviation from the average exceeds the threshold before the data is considered as suspect. First, if the point is on a continuously increasing or decreasing series (i.e., points for which $y_{i-1}<y_i<y_{i+1}$ or $y_{i-1}>y_i>y_{i+1}$), then these points are not checked, since these data sequences are realistic. Second, two lines are fit through the four points either side of the suspect points, and the average predicted value at the position in question from these two lines is calculated. If this value is further from the average than the measured value, then it is assumed that there is a very steep peak in the data, and that the measured value is realistic.

The points with the suspect values are then corrected in order of decreasing deviation. The procedure for correcting the data is to fit a 4^{th} order polynomial through the 20 data points

around the suspect point (excluding the suspect data point itself, and any data points around it that are considered suspect). The equation for this curve can be expressed as:

$$\begin{bmatrix} \sum_j 1 & \sum_j x_j & \sum_j x_j^2 & \sum_j x_j^3 & \sum_j x_j^4 \\ \sum_j x_j & \sum_j x_j^2 & \sum_j x_j^3 & \sum_j x_j^4 & \sum_j x_j^5 \\ \sum_j x_j^2 & \sum_j x_j^3 & \sum_j x_j^4 & \sum_j x_j^5 & \sum_j x_j^6 \\ \sum_j x_j^3 & \sum_j x_j^4 & \sum_j x_j^5 & \sum_j x_j^6 & \sum_j x_j^7 \\ \sum_j x_j^4 & \sum_j x_j^5 & \sum_j x_j^6 & \sum_j x_j^7 & \sum_j x_j^8 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ e \\ f \end{bmatrix} = \begin{bmatrix} \sum_j y_j \\ \sum_j y_j x_j \\ \sum_j y_j x_j^2 \\ \sum_j y_j x_j^3 \\ \sum_j y_j x_j^4 \end{bmatrix}$$

$$j = \{i-10, \ldots, i-1, i+1, \ldots, i+10\} \qquad \textbf{(3)}$$

$$\hat{y}_i = a + bx_i + cx_i^2 + dx_i^3 + ex_i^4$$

Before the suspect data value is replaced with the approximation above, a number of checks are performed. However, these are very difficult to describe, and the reader is referred to the spreadsheet formulae for the exact checks. First, if the approximation has a greater deviation from the eight point average than the suspect value, then the data is not altered. In some cases, these suspect data values occur at large jumps in the data, rather than at single spikes. These are caused by loose connections—the operators should repeat the readings, but sometimes these data slips through. If there is a large jump in the data, then the 4th order approximation is not appropriate, and the predicated value from a straight line through the ten points on the side of the data closer to the average is used. Also, if the suspect point is within 10 points of either end of the data this one sided linear approximation is used.

This procedure is shown graphically in Figure 13, although a better understanding of the algorithm can only be gained by examining the implementation in the loader and watching as it processes data.

Finally, a check is performed to determine if the suspect value (y_i) and the new approximate value differ by more than ε. If they do not, the point is skipped in order to prevent the algorithm from looping indefinitely. Once the value has been replaced, all of the values are recalculated and the procedure is repeated until no deviations exceed the threshold. This cleaning method is designed to touch as few data points as possible while completely removing the suspect values from the data set.

On most (>99 percent) of the existing bowls the procedure works very well, not altering data that visually appears to have no problems, and only altering a few (<10) data points on bowls that have obvious spikes or other spurious measurements. If the data is extremely noisy, then the procedure attempts, but normally will not be successful in producing a realistic corrected bowl. There is not much that can be done about this, other than trying to limit this data, and repeating measurements that are noisy.

4.2.3 Data smoothing

In some cases, the data not only needs to be cleaned, but the variance of the data needs to be reduced. This variance may be caused by noise in the DAS or by randomness in the measurements. To smooth the data, a simple moving average approach is used. This approach appears to work well, provided the data has been cleaned first. The reason that the data has to be cleaned first is because if spikes or other spurious points exist in the data, then the moving average propagates these values into the adjacent data.

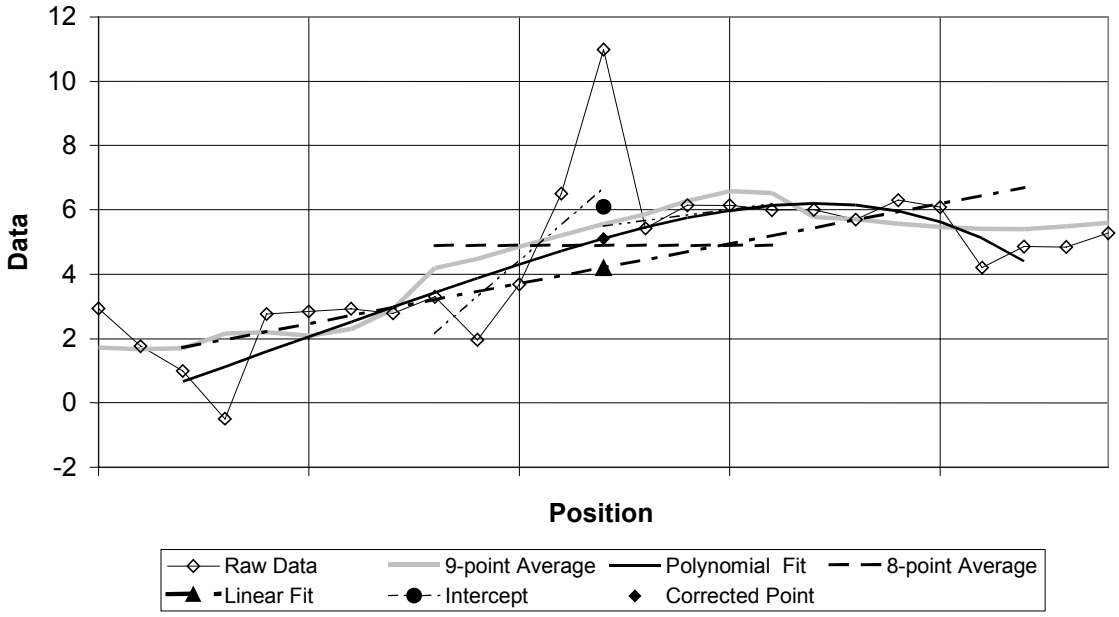


Figure 13. Example of data cleaning

There are three different types of moving average used at various stages in the data analysis. The first is a weighted average of five points (Equation 4a), the second a straight average of five points (Equation 4b), and the third is a straight average of nine points (Equation 4c).

$$\hat{y}_i = \frac{y_{i-2} + 3y_{i-1} + 5y_i + 3y_{i+1} + y_{i+2}}{13} \quad \textbf{(4a)}$$

$$\hat{y}_i = \frac{y_{i-2} + y_{i-1} + y_i + y_{i+1} + y_{i+2}}{5} \quad \textbf{(4b)}$$

$$\hat{y}_i = \sum_{j=i-4}^{i+4} y_j \Big/ 9 \quad \textbf{(4c)}$$

As can be seen in Figure 14, the filters all remove a significant amount of the local variability in the data while not affecting the general trend over even a fairly small number of points.

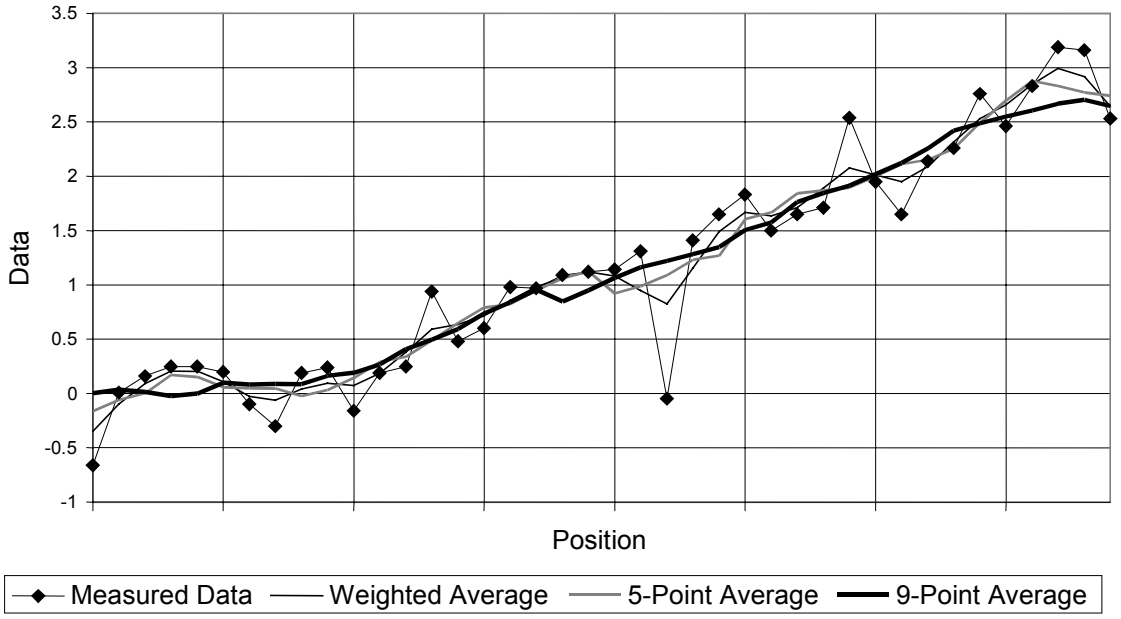


Figure 14. Example of smoothing filters.

4.2.4 Inversion of deflection bowls

Generally, deflection data should show the pavement deflecting downwards. However, the instruments are often calibrated so that the pavement appears to deflect upwards. The loader allows for an instrument orientation of "-Z" to always correct instruments for which this is true for the entire test. However, sometimes there are inverted bowls, especially for the RSD.

To check for these cases, a fairly simple algorithm is applied. The standard deviation of the first and second half of the data is used to determine which side of the data represents the "flat" part of the bowl and which the "peak." The average of the end 10 percent of the data on the "flat" side is then used as the "zero" for the instrument. The maximum and minimum deviation from this "zero" are then calculated and used to determine if the bowl is inverted. If the upward deflection exceeds the downward deflection, then the bowl is inverted. However, because the bowl may be noisy, just using a single point maximum or minimum to check this is prone to error, and so the 5^{th} and 95^{th} percentile values of y are used in place of the minimum or

maximum. Also, if the total deviation is less than 2σ (from the cleaning algorithm) then it is assumed that bowl is almost flat and it's not possible to determine if it is really inverted.

Inverted RSD data are not a major problem. To invert the data, they can simply be multiplied by –1, since only the peak deflection is of interest. However, inverted MDD or JDMD data requires that the permanent deformation values for these instruments be corrected. This is not possible in an automated fashion, and so these data bowls are discarded if they are inverted. An orientation of "–*Z*" can and should be used to correct this data while it is being loaded.

4.3 Level One Data Analysis

Pavement response data is either a deflection, a strain, or a stress. For these types of data, two summary variables are of primary interest: the current static value of the parameter and the deviation caused by dynamic loading. For some instruments, other summary variables that provide some indication of how the pavement is reacting to loading are also often presented in level one data analysis reports. The only instrument that doesn't conform to this pattern is the profilometer, since it does not involve any loading.

4.3.1 Deflection response processing

The first stage in processing the data is to establish the subset of the data to calculate the static deflection. Since most pavement deflection data is collected with the wheel starting far from the instrument, it could be assumed that the first data points are the best for calculating the static deflection, however, this is not always true. Because of this, an algorithm for determining which direction the wheel was moving was established, and where the wheel would have been in relation to the pavement and thus the instrument. Once this has been established, it can be used

to determine the side of the deflection bowl that is furthest from the instrument and a "pd range," which begins one point from the end of the data (the first or last point is excluded because these are often noisy and difficult to correct in the cleaning algorithm). The "pd range" covers the greater of either 20 points or 200 mm. If the data set is for a static test, then the entire data set, excluding the first and last points is used as the "pd range." The average value over this range is determined as static reading for the instrument, denoted as y_0. If the data is for a static measurement, then the calculations are complete, and the data is ready to be loaded into the database.

For the old DOS-based DAS, the x_i values are initially just relative measures of wheel location, and they need to be shifted so that the x_i values are in the same coordinate system as the instrument. To adjust the values, the location of the instrument in relation to the x_i values needs to be determined. To do this, it is assumed that the instrument's peak dynamic deflection is when the wheel is directly over the instrument. Once again, using only the single point of maximum deflection is error prone, and so parabola is fit to the 21 data points around the location of the peak (similar to Equation (3)) and the turning point (x_{peak} =-b/2c) is calculated. Since the location of the instrument within the coordinate system of the section is known, the **x** values can be translated into this coordinate space:

$$x_i' = x_i - x_{peak} + x_{instrument}' \quad \textbf{(5)}$$

Figure 15 shows the calculation of the permanent deformation and peak deflection (in this case for a JDMD bowl). As can be seen, there is a small error in the instrument location because of the sharp change in the deflection as the wheel crosses the joint—this is acceptable because the JDMD would not be located exactly at the joint.

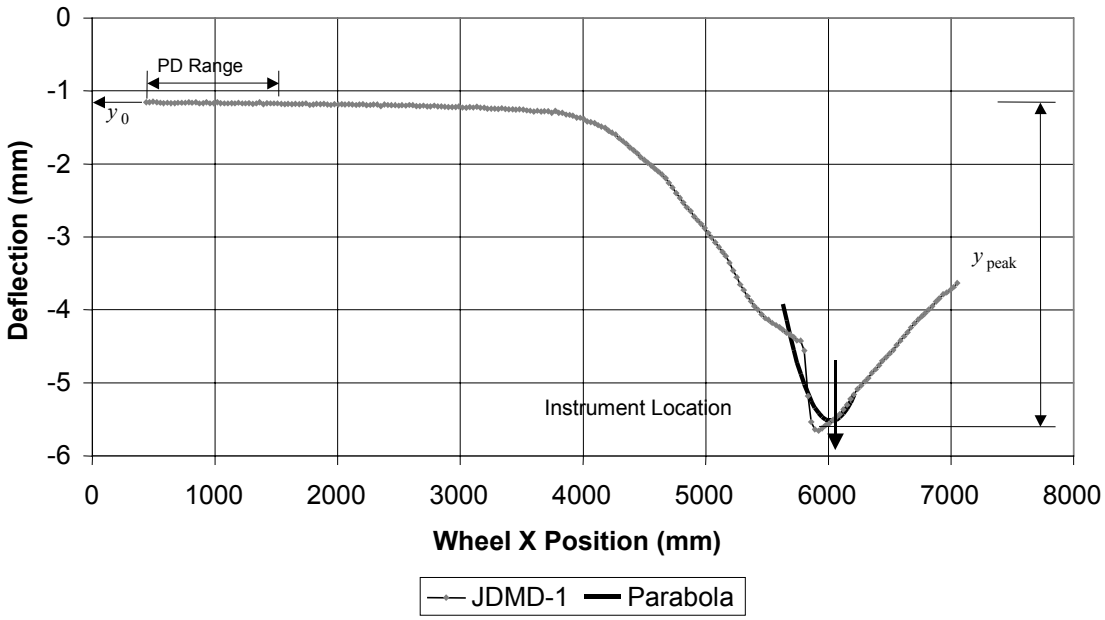


Figure 15. Example deflection bowl (for JDMD).

To simplify the data queries on the raw data in the database, the new positions are rounded to the nearest multiple of the distance between readings.

4.3.2 Load Transfer Efficiency

One special parameter which needs to be calculated is the Load Transfer Efficiency (LTE) of joints in a concrete pavement. The true value of this parameter cannot be calculated, because it would require the measurement of loads not deflections, but it can be estimated using the deflections from two JDMDs, one on either side of the joint. An example is shown in Figure 16.

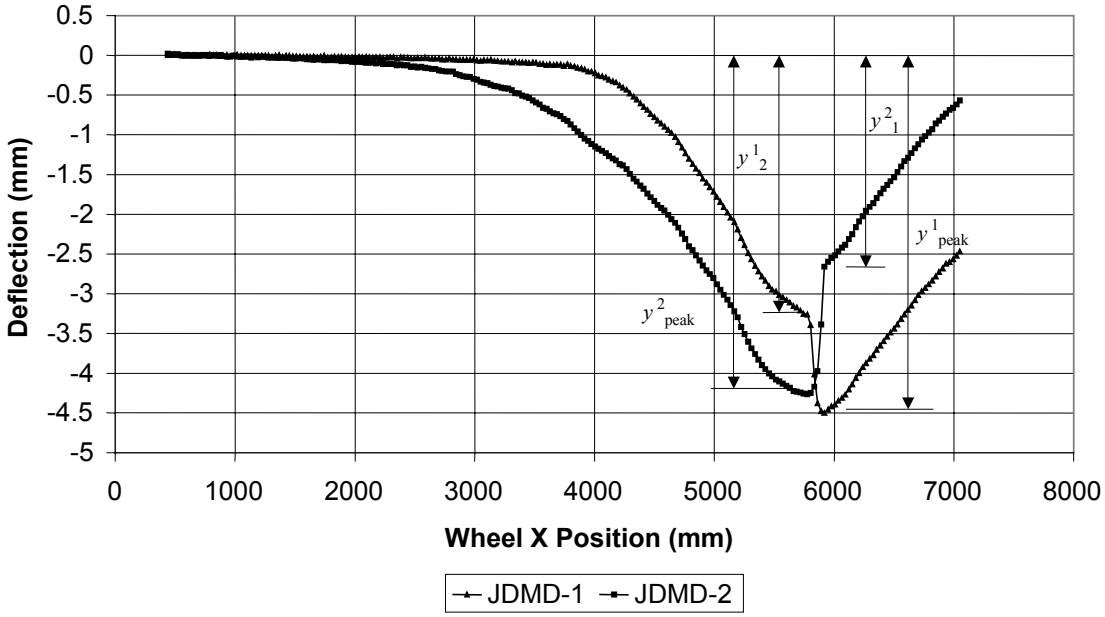


Figure 16. Example of LTE calculation.

There are many different ways of defining LTE, but two possible calculations (per instrument) can be performed from the four data values above:

$$LTE^1 = \frac{y_2^1}{y_{peak}^2} \quad \textbf{(6a)}$$

$$LTE^1 = \frac{y_2^1}{y_{peak}^1} \quad \textbf{(7b)}$$

The second definition (Equation 6b) has been implemented in the PRC HVS database because it is more stable (since it only uses deflections on the scale of instrument 1). The first (Equation 6a) is closer to the definition used for obtaining an approximation from FWD data, but can generate values greater than 100 percent, which are difficult to explain. Both approximations of LTE can give negative results. This implies that the one slab is lifting while the load is on the other slab, and might indicate pumping, or a very badly cracked section.

4.3.3 Profilometer Data Analysis

The data collected from a profilometer differs from the other kinds of data because it is a set of deformation data, and must always be related to the initial profile determined for a pavement. The profiles also need to be leveled, since the profilometer measures the true profile of the pavement, including any crossfall (assuming that the profilometer is level). To do this, a pseudo zero is calculated on either side of the section, from 30 data points located at least one third of the section width outside of the section. The pseudo zeros are used to calculate two zero points, and a line is fitted through these points, and the deviations from this line calculated. Once the leveled profile and leveled initial profile have been determined, the deformation is determined by:

$$rut_i = y_i - \frac{1}{9}\sum_{j=i-4}^{i+4} zero_i \qquad \textbf{(8)}$$

This is shown in Figure 17, along with the smoothed profile (using the weighted average of five points). The reason for using a moving average from the initial profile is because the profilometer also detects surface micro texture, which should be excluded from the calculation of the rut depth.

Once the profile has been determined, the rut depth can also be determined. The rut depths are only calculated over a width from one third of the section width on either side of the section itself. Four rut depths are calculated: the maximum downwards deformation from the original profile (*maximum deformation*), the maximum upwards deformation (*minimum deformation*), the maximum difference between the maximum and minimum (*maximum rut*) and the average deformation within the section width (*average deformation*).

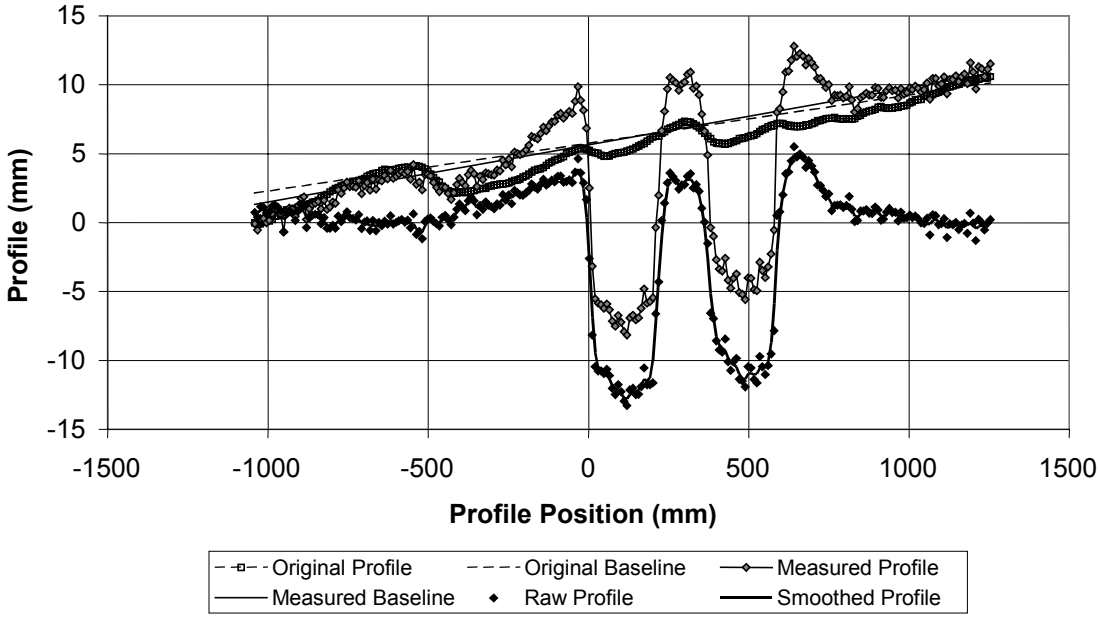


Figure 17. Profile calculation.

The determination of the maximum difference depends on whether the section has wandering traffic or channelized traffic. For wandering traffic (or single wheels), it is merely the difference between the maximum deformations and the minimum deformations. For channelized dual wheel traffic, it is determined as the maximum or rut_1 through rut_4 from Figure 18.

To obtain the the maximums and minimums, the locations of the maximums are first determined in the first and second halves of the section. The locations of the three minimums are then determined between the edge of the calculation width and the maximums, and between the maximums.

Because of the smoothing of the data in Equation (8), the first profile recorded on the section does not have zero rut—it is normally of the order of 3 mm. The rut depth of this section represents the surface texture of the pavement, and is subtracted from the subsequent results, to determine the real deformation of the structure.

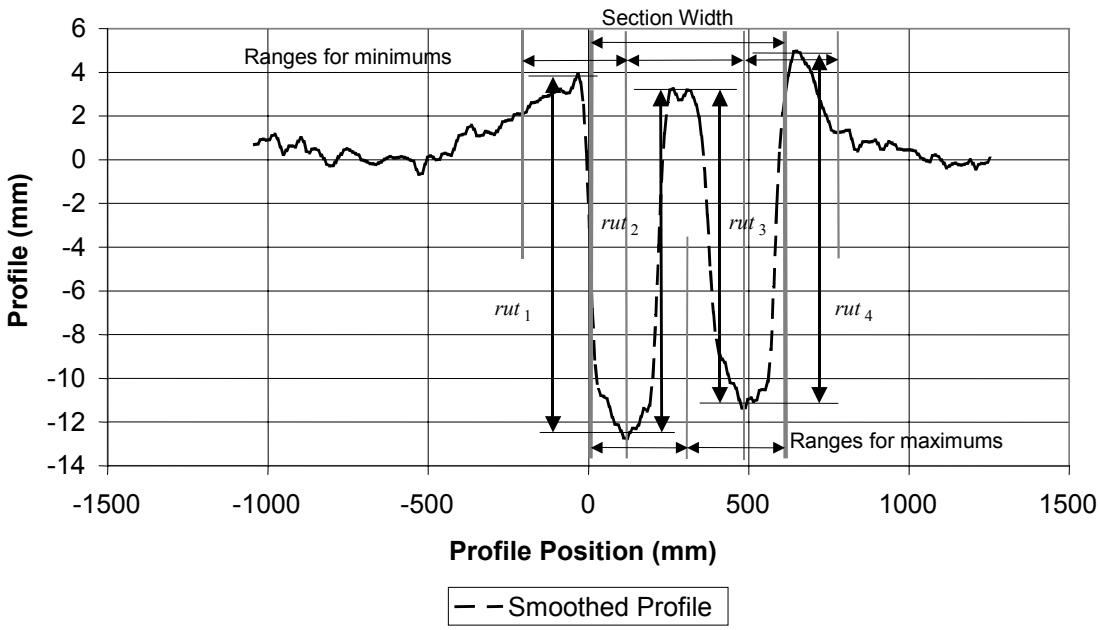


Figure 18. Determination of rut depths.

5.0 CONCLUSIONS

Because of changes in the way that HVS testing has been performed for the development of the Caltrans Mechanistic-Empirical Design Method, a new HVS database had to be developed. This database, known as the PRC-HVS database, contains all of the HVS data collected from both HVS's since 1995. Data from both rigid and flexible pavements are included in the same database, since most of the instrumentation operates in a similar fashion.

The database structure enforces many relationships to ensure that the data in the database is consistent. A number of additional queries are also performed to ensure that the data that does not fit well into the relational structure of the database is also consistent.

Although the database is operational, there are a number of improvements that can still be made. One of the main enhancements to database will be to begin linking in the long term pavement performance data that has been collected as part of the project along to the laboratory

testing on the materials for the various test section. With the additional information from these other databases, it will be possible to implement some automated level two data analysis within the database.