Hochschule Esslingen
University of Applied Sciences

Exercises in Databases

Part 1: SQL

Prof. Nonnast

Version 4.9

last update 2020-07-29

1. EXPLORE YOUR DATABASE....3

2. PROJECTION, SELECTION....4

3. SUB SELECTS WITH IN, EXISTS, ANY, ALL....7

4. DATE FUNCTIONS....1

5. AGGREGATE FUNCTIONS....2

6. CREATE TABLES....3

7. EQUI-JOIN (RÉSUMÉS)....1

8. COMPLEX EQUI-JOIN (MEASUREMENT RESULTS AND FAULTS)....2

9. INNER AND OUTER JOIN (FAULT LISTINGS)....3

10. CROSS JOIN....4

11. EMBEDDED SQL....1

APPENDIX A QUALITY INFORMATION SYSTEM....1

APPENDIX B DESCRIPTION OF USED ATTRIBUTES....1

1. Explore your Database

DB2

'LIST TABLES FOR ALL' gives out all user tables and synonyms. Compare tables defined in the database with entities defined in the given Entity Relationship Model.

'LIST TABLES FOR SCHEMA labor; gives you all tables owned by the connected user. Remember you didn't define any table up to now.

'DESCRIBE TABLE <table_name>' gives you the definition of a table. For long column names the database system will truncate column headings. In this case the DESCRIBE command will give you the full column name.

Compare the column names of the database tables headers and resumes with the attribute names of the corresponding entities in the Entity Relationship Model.

'VALUES (CURRENT SCHEMA)' gives you the name of the current schema. Try also CURRENT_SERVER, CURRENT_USER, CURRENT_DATE, CURRENT_TIME!

Command Line Processor (CLP) options can be set by using DB2OPTIONS environment variable. For more details using command line processor see command reference manual, chapter 2. Further important data dictionary tables are listed in Appendix E .

Some rules for a code of high quality

- Use capital letters for keywords to get a user-friendly code.
- Always add an ORDER-BY-clause.
- Do not skip keywords. The DBMS will add defaults, but nobody knows them by heart, e.g. ASC.
- Qualify your JOIN by INNER, OUTER, CROSS, … .
- Omit superfluous keywords, e.g. DISTINCT for a key-based search.

2. Projection, Selection

2.1. Which engine (cubic capacity) goes with which series? Sort your output to the given order!

```
CAPACITY   SERIES
---------- ----------------
320        C-Kl
280        C-Kl
240        C-Kl
430        E-Kl
320        E-Kl
280        E-Kl
500        S-Kl

7 rows selected.
```

2.2. The ok_not_ok state of an engine is coded either by 0 or by 1. Select the engines whose state is 1? Sort your output alphabetically by aggregate_id.

```
AGGREGATE_ID    OK_NOT_OK
--------------- ------------
MA  0197770     1
MA  0231566     1
MA  0250274     1
MA  0250287     1
MA  0330010     1
XL  0026065     1
XL  0033027     1

7 rows selected.
```

2.3. Rename the column AGGREGATE_ID to ENGINE!

```
ENGINE          OK_NOT_OK
--------------- -------------
MA  0197770     1
MA  0231566     1
MA  0250274     1
MA  0250287     1
MA  0330010     1
XL  0026065     1
XL  0033027     1

7 rows selected.
```

2.4. Continuing exercise 2.3 replace the value 1 by OK and rename column header to IO!

```
ENGINE          IO
--------------- --
MA  0197770     OK
MA  0231566     OK
MA  0250274     OK
MA  0250287     OK
MA  0330010     OK
XL  0026065     OK
XL  0033027     OK

7 rows selected.
```

2.5. For assembling an engine many assembling stations are needed. Give all stations which the engines ‘XL 0026065’ and ‘MA 0197770’ passed during their assembly process. Order your output chronologically to the date of storage. So, you will get résumés of the production of these engines.

Hint:

For an aggregate_id notice the two blanks following the two first characters.

```
ENGINE          STATI  NO STATION_LEAVING_DATE
--------------- ----- --- --------------------
MA  0197770     KTS2    5 30.04.2001 11:51:57
MA  0197770     KTS4    4 30.04.2001 10:39:53
MA  0197770     KTS4    3 30.04.2001 10:36:26
MA  0197770     KTS4    2 30.04.2001 09:45:13
MA  0197770     KTS4    1 30.04.2001 09:42:14
XL  0026065     KTS6    5 17.09.2001 10:34:22
XL  0026065     KTS7    4 16.09.2001 21:53:20
XL  0026065     KTS7    3 16.09.2001 21:52:59
XL  0026065     KTS7    2 16.09.2001 21:51:20
XL  0026065     KTS7    1 16.09.2001 21:51:08

10 rows selected.
```

2.6. Because of a defective measuring receiver, results (MEASURED_VALUE) produced by some testing machines are not useful. Such results exceed the upper limit by 7 % or go more than 7 % below the lower limit. Look for such useless results. Restrict your output to the engine 'XL 0026065'. (Hint: You need no functions. Simple arithmetic will do the job.)

```
AGGREGATE_ID    LOWER  M_VALUE UPPER
--------------- ------ ------- ------
XL  0026065       -480    -605   -345
XL  0026065       -480    -636   -345
XL  0026065       -240    -405    220
XL  0026065       -240    -449    220
XL  0026065         12       5     50
XL  0026065         12       5     50
XL  0026065         12       7     50
XL  0026065         20      16     25
XL  0026065         20      15     25
XL  0026065         20      12    100
XL  0026065         20      13    100
XL  0026065         20      13    100
XL  0026065         50       3    200
XL  0026065         50       4    200
XL  0026065         80       4    120
XL  0026065         80       1    120
XL  0026065        100      58    300
XL  0026065        100      54    300

  18 record(s) selected.
```

3. Sub selects with IN, EXISTS, ANY, ALL

3.1. Table station_descriptions stores all existing assembly machines. Look for the full name of all the machines referenced in table resumes!

```
ID    STATION_TEXT
----- -----------------
HTS3  Hot test station 3
KTS2  Cold test station 2
KTS3  Cold test station 3
KTS4  Cold test station 4
KTS5  Cold test station 5
KTS6  Cold test station 6
KTS7  Cold test station 7

7 rows selected.
```

3.2. List all defined errors stored in table error log! For each error code add its corresponding error text!

```
COD ERROR_TEXT
--- --------------------
IO  kein Fehler (Montage)
NIO NIO

2 rows selected.
```

3.3. Communication between machines and database system became faulty. It may be possible that some of the rows stored in table error log are corrupt. You are sure that the youngest entry of engine ‘XL 0033027’ is correct. Rows stored later may be corrupt. Give all the corrupted rows.

```
AGGREGATE_ID    STATI ERR ERROR_REPORTING_DATE
--------------- ----- --- -------------------
MA  0330010     B250  IO  29.10.2001 05:39:26
MA  0330010     B270  IO  29.10.2001 05:53:19
MA  0330010     B350  IO  29.10.2001 07:42:11
MA  0330010     M220  IO  29.10.2001 05:46:42
MA  0330010     M461  IO  29.10.2001 08:44:57
MA  0330010     M467  IO  29.10.2001 08:54:23
MA  0330010     M542  IO  29.10.2001 09:45:36
MA  0330010     M547  IO  29.10.2001 09:53:47
MA  0330010     M556  IO  29.10.2001 10:08:25
MA  0330010     A450  NIO 29.10.2001 08:15:58

10 rows selected.
```

4. Date functions

4.1. How many hours did it take to assemble each engine? Duration is the difference between end of assembly process and start of assembly process.

Hint:

```
DB2:    75301 to be interpreted as 7h 53min 01sec; try TIMESTAMPDIFF()
Oracle: 7.88361111 as Floating Point
```

```
CAPACITY   SERIES     DURATION
---------- ---------- ------------
320        C-Kl              7.88
320        C-Kl             16.50
280        C-Kl              6.73
240        C-Kl              9.57
430        E-Kl              8.67
320        E-Kl             20.63
280        E-Kl              8.38
500        S-Kl             26.00

8 rows selected.
```

4.2. In exercise 2.5 you already got the résumé for each engine. Now give résumé entries that belong to early terms only. Mark these rows with Shift A. Shift work consists of three terms, the early term, the late term and the night term. Early term begins at 6 am and lasts until 2.30 pm.

```
ENGINE          STATI  NO STATION_LEAVING_DATE   SHIFT
--------------- ----- --- -------------------- -------
XL  0026065     KTS6    5 17.09.2001 10:34:22  Shift A
MA  0197770     KTS2    5 30.04.2001 11:51:57  Shift A
MA  0197770     KTS4    1 30.04.2001 09:42:14  Shift A
MA  0197770     KTS4    2 30.04.2001 09:45:13  Shift A
MA  0197770     KTS4    3 30.04.2001 10:36:26  Shift A
MA  0197770     KTS4    4 30.04.2001 10:39:53  Shift A

6 rows selected.
```

Hint for DB2:

Use the function TIME (<expression>). The TIME function returns a time from a value. The argument must be a time, timestamp, or a valid character string representation of a time or timestamp. The result of the function is a time. If the argument can be null, the result can be null; if the argument is null, the result is the null value.

5. Aggregate functions

5.1. Look for the maximum and the minimum sequence number. What are the generic names for the column headings given by the database system? Use the column headings given below.

```
LOWEST      HIGHEST     COUNT
----------- ----------- -----------
          1           5          22

1 rows selected.
```

5.2. Look for the highest and the lowest sequence number of each engine.

```
ENGINE          LOWEST      HIGHEST     COUNT
--------------- ----------- ----------- -----------
MA  0033093               1           1           1
MA  0197770               1           5           5
MA  0231566               1           2           2
MA  0250274               1           2           2
MA  0250287               1           2           3
MA  0330010               1           3           3
XL  0026065               1           5           5
XL  0033027               1           1           1

8 rows selected.
```

5.3. For data being consistent one condition could be the following: no résumé entry is missing. Compare the number of rows belonging to an individual engine with the highest sequence number. (In real life consistency is much more complex). Select all engines being not consistent.

```
ENGINE          LOWEST     HIGHEST   COUNT
--------------- ---------- --------- ---------
MA  0250287              1         2         3

1 rows selected.
```

5.4. Check your result from exercise 5.3 by selecting the complementary set of rows. Restrict your output to all engines beginning with the string MA.

```
ENGINE          LOWEST     HIGHEST   COUNT
--------------- ---------- --------- ---------
MA  0033093              1         1         1
MA  0197770              1         5         5
MA  0231566              1         2         2
MA  0250274              1         2         2
MA  0330010              1         3         3

5 rows selected.
```

5.5. In looking for consistent résumés entries in exercise 5.4 ignore all entries coming from stations whose station_id starts with 'HTS'. Even though the where-clause is now more restrictive than in exercise 5.4 the result set in this exercise 5.5 has more rows than the less restrictive exercise 5.5. Find an explanation based on aggregate_id 'MA 0250287'!

```
ENGINE          LOWEST      HIGHEST     COUNT
--------------- ----------- ----------- -----------
MA  0033093               1           1           1
MA  0197770               1           5           5
MA  0231566               1           2           2
MA  0250274               1           2           2
MA  0250287               1           1           1
MA  0330010               1           3           3

6 rows selected.
```

5.6. Get the number of engines assembled per year! As some DBMS do not support functions in a group by clause try to use the YEAR() function outside the group by clause.

```
    COUNT TEXT             YEAR
--------- ---------------  ----
        1 Engines per Year 2000
        7 Engines per Year 2001

2 rows selected.
```

6. Create tables

Hint: Verify your data immediately after you manipulated them.

6.1. Save the result of exercise 4.2 in a new table shift! Create this table in YOUR OWN SCHEMA, since you do not have privilege in schema labor.

6.2. Add the following row into table shift!

```
XL  0197779 KTS4    4 15.09.2001 22:39:52 Shift B
```

6.3. Set the attribute station_leaving_date to the current date and time for all rows with sequence_no = 4. Use the appropriate system variable!

6.4. Empty table shift! Is there really no more row in table shift?

6.5. Destroy table shift!

6.6. What is the difference between both commands DROP and DELETE?

7. Equi-Join (résumés)

7.1. Which assembling stations does engine ‘XL 0026065’ pass through during its assembling process? Get the rows ordered chronologically!

```
ENGINE          STATION_TEXT         NO  STATION_LEAVING_DATE
--------------- -------------------- --- --------------------
XL  0026065     Cold test station 7      1 16.09.2001 21:51:08
XL  0026065     Cold test station 7      2 16.09.2001 21:51:20
XL  0026065     Cold test station 7      3 16.09.2001 21:52:59
XL  0026065     Cold test station 7      4 16.09.2001 21:53:20
XL  0026065     Cold test station 6      5 17.09.2001 10:34:22

5 rows selected.
```

7.2. Complete the output given in exercise 7.1 with information about begin (assembly_start_date) and end of the assembling process (assembly_end_date) of engine ‘XL 0026065’.

Think about the output as a unification of three result sets.

```
ENGINE          STATION_TEXT         NO  STATION_LEAVING_DATE
--------------- -------------------- --- -------------------
XL  0026065     Assembly began         0 16.09.2001 14:34:51
XL  0026065     Cold test station 7    1 16.09.2001 21:51:08
XL  0026065     Cold test station 7    2 16.09.2001 21:51:20
XL  0026065     Cold test station 7    3 16.09.2001 21:52:59
XL  0026065     Cold test station 7    4 16.09.2001 21:53:20
XL  0026065     Cold test station 6    5 17.09.2001 10:34:22
XL  0026065     Assembly finshed       0 17.09.2001 16:34:38

7 rows selected.
```

8. Complex Equi-Join (measurement results and faults)

8.1. For which measured values does a corresponding error exist.

```
AGGREGATE_ID    FEATURE_TEXT              STATION_TEXT          NO  ERR
--------------- ------------------------- -------------------- --- ---
MA  0250287     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
MA  0250287     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
MA  0250287     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
MA  0250287     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO
MA  0250287     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO
MA  0330010     Hautbildungszeit          A450:  Zylinderkopf v  1 NIO
XL  0026065     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
XL  0026065     Differenz zu LR Shortbloc A1510: Lecktest Long   1 NIO
XL  0026065     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
XL  0026065     Leckrate Ol (Prufdr. 0.25 A1510: Lecktest Long   1 NIO
XL  0026065     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO
XL  0026065     Leckrate Wasser (Prufdr 1 A1510: Lecktest Long   1 NIO

19 rows selected.
```

9. Inner and Outer Join (Fault listings)

9.1. Which errors did occur? Give a summary.

```
ERROR_TEXT                          COUNT
---------------------------------- ---------
No fault (Asembly)                         64
NIO                                        10

2 rows selected.
```

9.2. List also errors which might had occurred. Such errors got the number 0 in the output.

Use a solution with outer join.

```
ERROR_TEXT                           COUNT
---------------------------------- --------
Abbruch durch Benutzer                   .0
Abbruch durch Prufstand                  .0
Fehler 900 Kalttest                      .0
Fehler 901 Kalttest                      .0
Moment zu gross                          .0
Moment zu klein                          .0
NIO                                      10
Nicht bearbeitet                         .0
Winkel und Moment zu gro?                .0
Winkel zu gro? und Moment zu klein       .0
Winkel zu gross                          .0
Winkel zu klein                          .0
Winkel zu klein und Moment zu gro?       .0
Winkel zu klein und Moment zu klein      .0
kein Fehler (Montage)                    64
kein Fehler (Prufstand)                  .0

16 rows selected.
```

9.3. Think of the difference in using either COUNT(*) or COUNT(error_logs.error_code) or COUNT (error_catalog.error_code) for counting rows in exercise 9.2 .

a) Select all error texts from table error_logs. Save the result in a temporary table which has an additional column ‘count’ initialized with 0.

b) Insert all faults from table error_logs into your temporary table. Use your statement from exercise 9.2. !

c) Sum up all faults.

d) Give a textual explanation for the differences.

10. Cross Join

10.1. How much time does it take to assemble an engine? Give an average value. Complete your output by computing also the longest and the shortest duration. Add the number of considered rows and the standard deviation.

Your solution should be portable. Do not use nested aggregate functions as well as other functions shipped by your database system!

Standard Deviation is defined as:

$$s^2 = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2}{n}$$

```
SLOWEST   AVERAGE   FASTEST   COUNT     STDDEV
--------- --------- --------- --------- ---------
    26.00     13.05      6.73         8     44.60

1 rows selected
```

11. Embedded SQL

Use the skeleton host code files and add missing SQL-statement. Comments in the files will help you.

11.1. Singleton Select

Select exactly one row from table headers. Your program should ask the user for a valid aggregate_id.

For the following exercises consider these error scenarios:

- *no access to database because of an invalid password*
- *more than one row returned*
- *no rows found.*

Use implicit error handling first!

Use explicit error handling next!

What happens if you use explicit error handling without deactivating implicit error handling? Have a look on preprocessed source code files.

11.2. Cursor Select

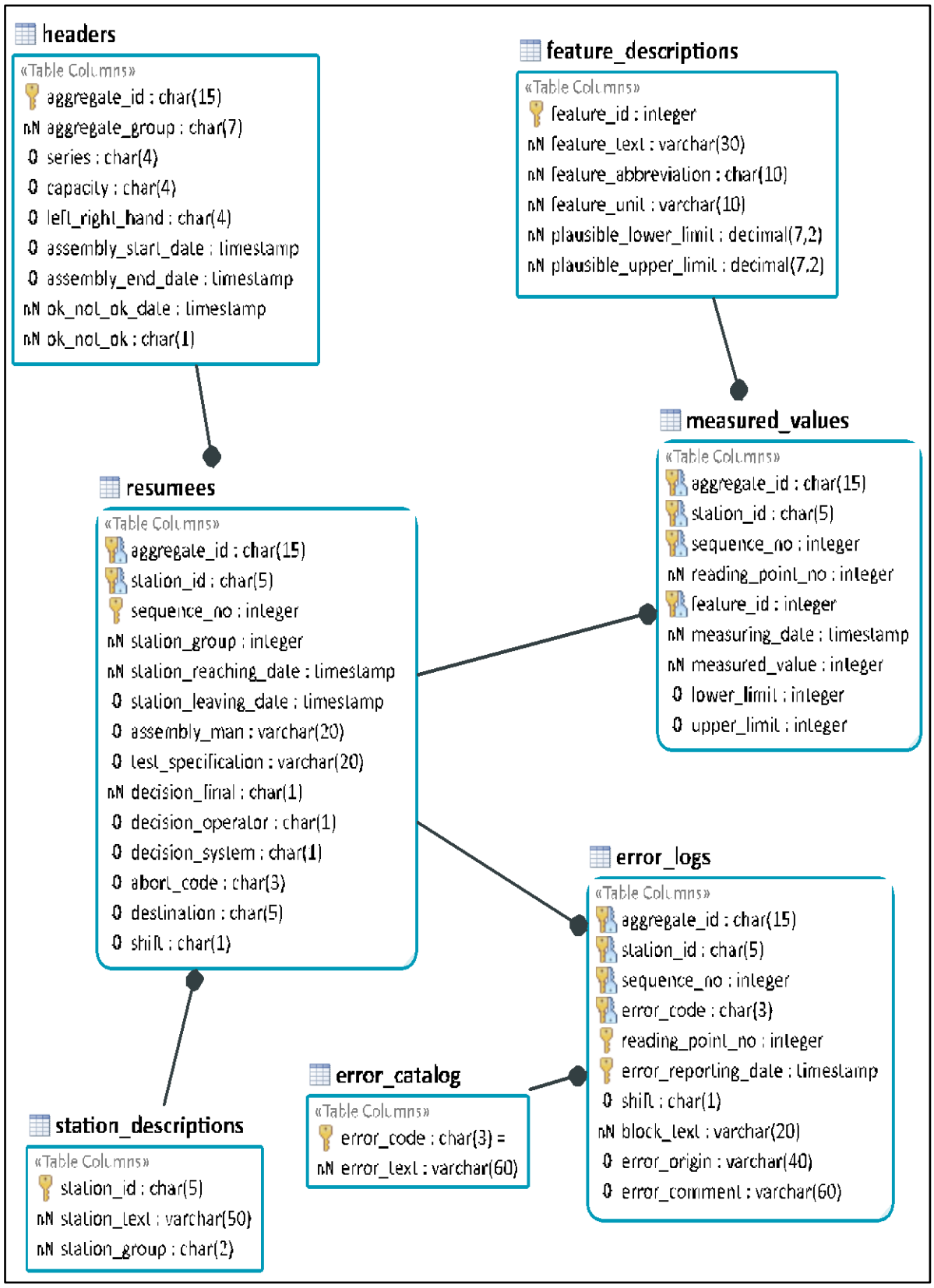
Optimize your source code from exercise 0 for returning more than one row. Your program should ask the user for a valid value of the ok_not_ok-flag.

11.3. Cursor Update

Select from table headers all rows belonging to series „E-Kl.“. Switch for all selected rows their ok_not_ok-flag from 0 to 1 and vice versa. Before updating a row, ask the user for confirmation. Do this for every selected row.

Verify your updates. Use interactive SQL. When will the data manipulation data be visible to other processes?

Appendix A Quality Information System



Appendix B Description of used attributes

Attribute	Description
abort_code	Encoded reason for breaking off an assembly step or a test suite.
aggregate_group	A group puts together aggregates, which are identical regarding assembly or testing.
aggregate_id	Identification number of an engine forced onto the engine block.
assembly_end_date	Assembly ends after the engine having finished final test.
assembly_man	Assembly man or testing operator of a unit.
assembly_start_date	Assembly starts at the beginning of the assembly line when the identification number is forced onto the engine block.
block_text	Code identifying the source of an error message. Possible values are “F” for repair center, “S” for testing operator, “M” for engine control. If there is any entry in this field further assembly of this individual engine is blocked.
capacity	Cubic capacity of an engine.
decision_final	Final judgment for an individual engine leaving a certain assembly or testing unit. Final judgment is based on a summary of all judgments, computer judgments and workman judgments, generated for this station. In case of any conflict workman judgments break computer judgments.
decision_operator	Judgment given by an assemblyman or testing operator based on manual tests.
decision_system	Computer-generated judgment based on results of automated testing.
destination	Identifying next station for further assembly.
error_code	Code, identifying an error occurred during assembly or testing. Possible codes and their description are listed in a catalog.
error_comment	Comment entered by assembly men.
error_reporting_date	Timestamp, when an assembly machine reported an error.
error_source	Who or what caused an error.
error_text	Description of an error code.
feature_abbreviation	Shortcut for a feature.
feature_id	Identification number of a feature. Features can be measurands or scheduled values.
feature_text	Description of a feature.

Attribute	Description
feature_unit	Unit of a feature, e.g. bar, °C.
left_right_hand	Assembly process differentiates between left-hand or right-hand traffic cars.
lower_limit	Legal lower limit for measured value.
measured_value	Scheduled values of measurement.
measuring_date	Timestamp, when the measured values were recorded. For a certain measurement all recorded measured values have the same timestamp.
ok_not_ok	Legal states are 0 or 1, representing an individual engine either being on the assembly line or being taken off from the assembly line for repairing reasons. Data packets are buffered and therefore may reach the database not in a chronological order. So, the stored state does not necessarily represent the actual state of an individual component.
ok_not_ok_date	Timestamp of the last switch of the state of an individual component.
plausible_lower limi	Feasible lower limit for values measured for a certain feature.
plausible_upper_limit	Feasible upper limit for values measured for a certain feature.
sequence_no	Counts how often an individual engine passed the same assembly or testing unit. The number is incremented by the DBMS when data is inserted into the database.
series	Name for a certain type of car given by manufacturer.
shift	Identifying the term when a certain error occurred. If work is organized in shifts then two or three terms, sometimes overlapping each other can be recognized.
station_group	A collective, which groups same types of assembly or testing units, e.g. “Hot test”, “Cold test”, “Automatic testing”, “Manual testing”.
station_id	Identifying an individual assembly or testing unit.
station_leaving_date	Timestamp, when an individual engine left an assembly unit.
station_reaching_date	Timestamp, when an individual engine reached a certain assembly unit.
station_text	Description of a certain assembly or testing unit, e.g. “piston assembly”. These units are often called stations.
test_specification	Identifying a test suite. Test suites often are realized as a computer program.
upper_limit	Legal upper limit for measured value.