Public API for Plansoft.org

Ref:PLAT/API/ 2013.08.06

Creator: Maciej Szymczak

Introduction

Audience for this document are developers and other technical stuff.

This document describes public API for plansoft, which allows you insert new classes into database, update existing classes and delete ones.

API simplifies integration process between PlanSoft and other planning systems.

API was implemented as a PL/SQL code within database.

In order to use it you need rights to connect to database.

Do not insert/update/delete records in database. Instead of this, use this API.

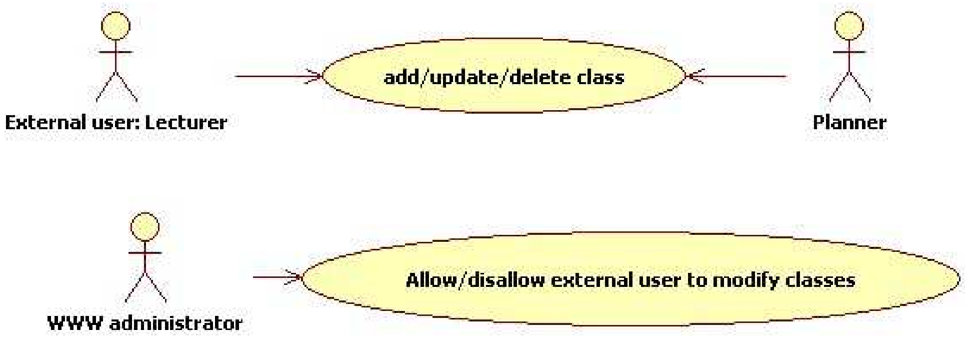
Scenerio

API gives possibility to plug-in external users into planning process.

Classes in PlanSoft are generally planned by specialized stuff.

It is assumed, that some classes can be preliminary planned by external users like lecturers, students and so on. External users have rights to plan in some period of time; access is controlled by www administrator.

External users interface is Internet Browser.



Whole plannig process consists of 5 steps:

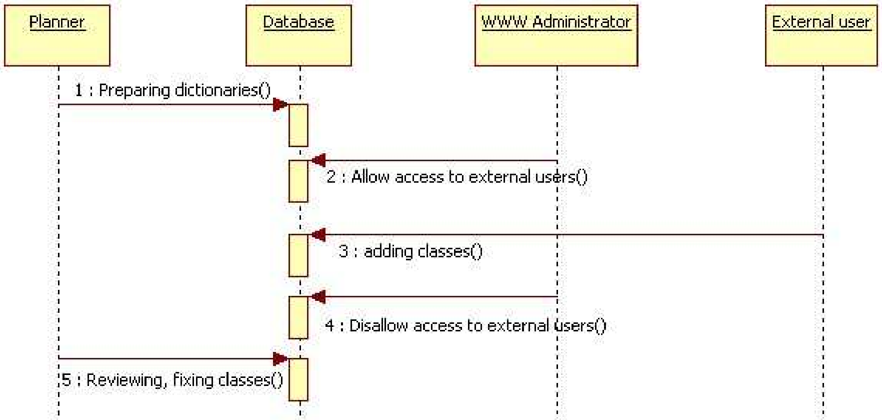
1/ Proffesional planner prepares dictionaries in a system like grups names, resources, subjects, forms, lecturer names.

2/ WWW administrator allows external users access to planning process

3/ External users are adding classes.

4/ WWW administrator disallows external users access to planning process

5/ Planner reviews plans, adds resources.


Planner
Database
WWW Administrator
External user
1 : Preparing dictionaries()
2 : Allow access to external users()
3 : adding classes()
4 : Disallow access to external users()
5 : Reviewing, fixing classes()

There are rules that have to be observed:

1. **At any moment in time either external user OR planners have to have access to planning classes. Otherwise chaos can occur.**

2. **External system should allow external users to change its own classes only (classes.created_by).**

3. **External system should limit resources available to external users, in particular: available dates and hours, available lecturers, groups, forms, resources, subjects.**

API Decription

API Package summary

```
  procedure insert_class     - Adds new class.
  function  can_insert_class- Checks if new class can be added. Does not add new class.
  procedure delete_class     - Deletes existing class.
  procedure update_class     - Updates class.

  function insert_class      - SQL equivalent for procedure insert_class (you can use this in SQL, no PLSQL is required).
  function update_class      - SQL equivalent for procedure update_class.
  function delete_class      - SQL equivalent for procedure delete_class.
```

Method	Description

procedure insert_class(
   papiversion number
  ,pday   date
  ,phour   number
  ,pfill   number
  ,psub_id   number
  ,pfor_id   number
  ,plec_ids   varchar2
  ,pgro_ids   varchar2
  ,prom_ids   varchar2
  ,powner   varchar2
  ,pcreator   varchar2
  ,out_class_id out number
 );
	Procedure adds a new class
	
	arguments:
	  papiversion API version. always set 1.
	  pday class date
	  phour class hour
	  pfill block utilization (1-100)
	  psub_id class subject (=subjects.id), see “DB Table Diagram”
	  pfor_id class form (=forms.id)
	  plec_ids class lecturer (=lecturers.id). this parameter is optional. You can pass more values separated by ";", for instance "1;2;3"
	  pgro_ids class group of students (=groups.id). this parameter is optional. You can pass more values separated by ";", for instance "1;2;3"
	  prom_ids class resource (=rooms.id). this parameter is optional. You can pass more values separated by ";", for instance "1;2;3"
	  powner =owner name, max. 30chars. it has to be the name of planner for instance BPLATA.
	  pcreator =creator name, max. 30chars. Any name uniquely indicated who actually created this class.
	  out_class_id id of just created class (=class.id)
	remarks:
	  it is assumed that dictionaries like subjects, forms, lecturers, groups, resources are given in database at the moment when you call this procedure.
	  this procedure grant automatically to owner permissions to see class in planner system.
	  this procedure adds creator into planners table.

	results: Any errors are reported by exception, examples: "ORA-00001: unique constraint (PLANNER.LEC_CLA_U) violated" means that you cannot add this class due to conflict with another one (Zaplanowanie tego zajêcia spodowa³oby konflikty z innymi zajêciami) This is frequently occured error "ORA-12899: value too large for column "PLANNER"."PLANNERS"."NAME" (actual: 71, maximum: 30)" - creator or owner name is too long "ORA-01722: invalid number" - values in plec_ids, ogro_ids, prom_ids are not separated by ";", but by another char "ORA-20002: Class without lecturer and group and resource" - (Wska¿ przynajmniej jeden z obiektow: wyk³adowcê, grupê lub salê) internal errors: "ORA-20000: API version error" - API version does not respond to external system version. "ORA-20001: Wrong owner name" - owner has to be user of type USER
function can_insert_class(papiversion number ,pday date ,phour number ,pfill number ,psub_id number ,pfor_id number ,plec_ids varchar2 ,pgro_ids varchar2 ,prom_ids varchar2 ,powner varchar2 ,pcreator varchar2) return varchar2;	Checks whether given class can be added, returns Y(insert is allowed) or exception is raised Procedure insert_class validates whether it is possible to add class as well. Thus you do not need to use this function. However, you may use it to inform user whether user action will be performed sucesfully in advance.
procedure delete_class (pid number);	Procedure deletes the class of given id arguments: pid class id (=classes.id) remarks: this procedure deletes class from database (grants to lecturer, group, resource, subject and form are not deleted. Creator in table planners is not deleted as well). THIS PROCEDURE DOES NOT CHECK WHETHER CURRENT USER HAS PERMISSIONS TO DO IT, IT IS A RESPONSIBILITY OF A SYSTEM WHICH USES THIS API. justification: External system functionality can differ from planner desktop functionality. For example, planner desktop allows changes to owner only (owner is a planner -

	employee,internal user). In opossite, external system can allow changes to creator rather than to owner (creator may NOT be a planner, it can be any external user for example a lecturer or a student). Usually the scenerio of plannig is: 1. First, external users plans its classes 2. Next, external users are disallowed to change data 3. Next, planner (insernal user) reviews plans, assigns rooms were missing and so on. It is important to implement appropriate safety rules on external system side, including period of time when external user are allowed to modifi data. results: Function returs no result. Any errors are reported by exception
procedure update_class(papiversion number ,pday date ,phour number ,pfill number ,psub_id number ,pfor_id number ,plec_ids varchar2 ,pgro_ids varchar2 ,prom_ids varchar2 ,powner varchar2 ,pcreator varchar2 ,in_out_class_id in out number);	Updates class. you need pass all elements of the class (NOT only changed elements) rememeber: updated class has another id (in_out_class_id).
function insert_class(papiversion number ,pday date ,phour number ,pfill number ,psub_id number ,pfor_id number ,plec_ids varchar2 ,pgro_ids varchar2 ,prom_ids varchar2 ,powner varchar2 ,pcreator varchar2) return number;	Function adds a new class Similar to procedure insert_class with the exceptions: - you can call this procedure in select statement (no pl/sql block is required) - this procedure performs commit sample code shows how to use this function. select api.insert_class(2 ,date'2010-06-04' -- pday ,'1' -- phour number ,100 -- pfill number ,(select max(id) from subjects) -- psub_id number

	,(select max(id) from forms) -- pfor_id number ,(select max(id) from lecturers) -- pcalc_lec_ids varchar2 ,(select max(id) from groups) -- pcalc_gro_ids varchar2 ,(select max(id) from rooms) -- pcalc_rom_ids varchar2 ,'BPLATA' -- powner varchar2 ,'WYKLADOWCA A' -- pcreator varchar2) result from dual results: function returns id of class
function update_class(papiversion number ,pday date ,phour number ,pfill number ,psub_id number ,pfor_id number ,plec_ids varchar2 ,pgro_ids varchar2 ,prom_ids varchar2 ,powner varchar2 ,pcreator varchar2 ,pclass_id in number) return number;	SQL version equivalent
function delete_class (pid number) return varchar2;	Almost the same as procedure delete_class. it allow you use this procedure directly in SQL (no PL/SQL block is required) this procedure does commit.

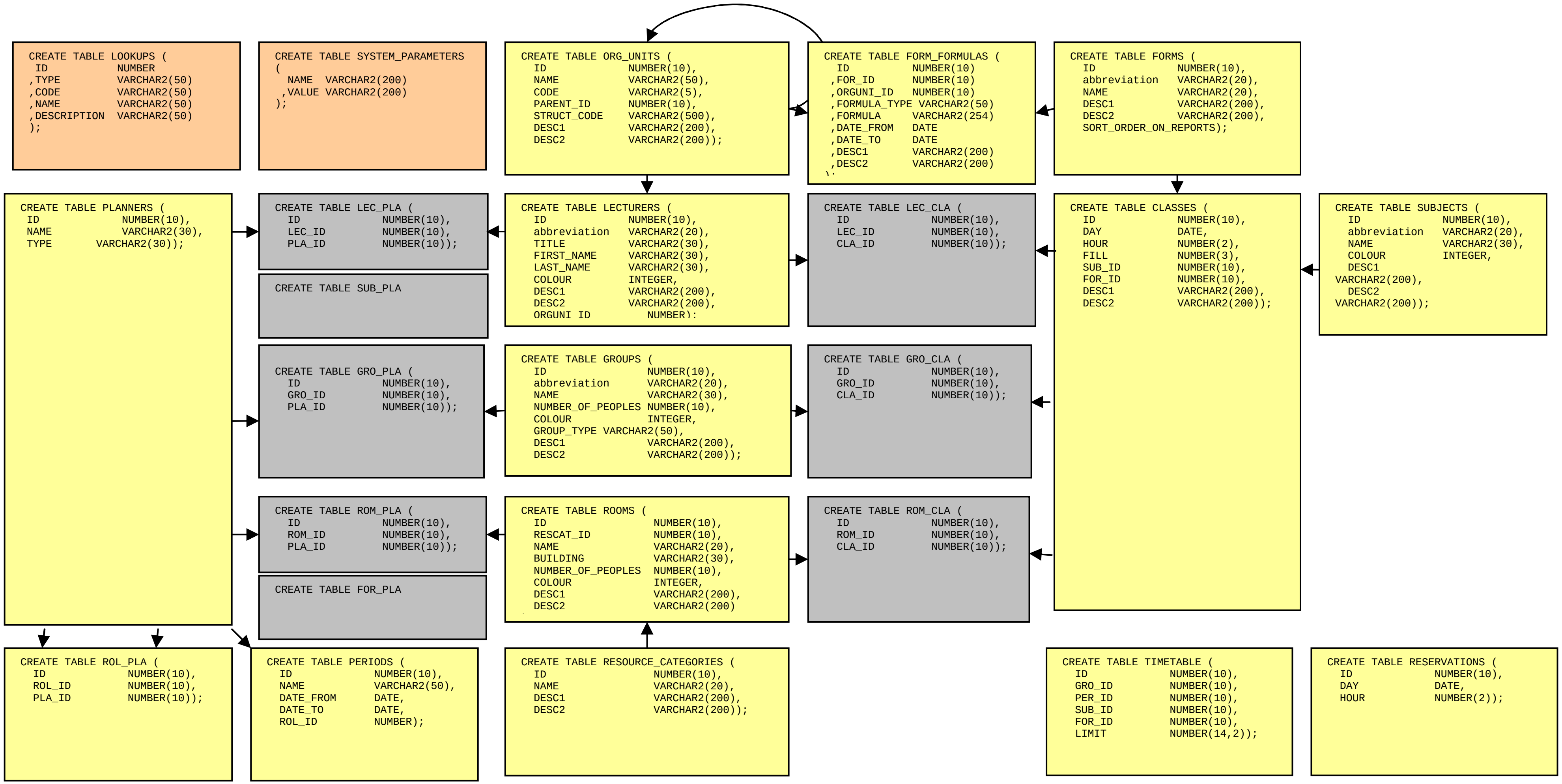
Sample script

```
Microsoft Windows XP [Wersja 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>sqlplus
```

```
SQL*Plus: Release 10.2.0.1.0 - Production on N Cze 6 08:33:53 2010

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Proszę podać nazwę użytkownika: planner
Proszę podać hasło:

Połączono z:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production


SQL> select api.insert_class(
  2        1
  3       ,date'2010-06-06'               -- pday
  4       ,'1'                            -- phour     number
  5       ,100                            -- pfill     number
  6       ,(select max(id) from subjects) -- psub_id   number
  7       ,(select max(id) from forms)    -- pfor_id   number
  8       ,(select max(id) from lecturers)
  9                                       -- pcalc_lec_ids   varchar2
 10       ,(select max(id) from groups)
 11                                       -- pcalc_gro_ids   varchar2
 12       ,(select max(id) from rooms)
 13                                       -- pcalc_rom_ids   varchar2
 14       ,'BPLATA'                       -- powner    varchar2
 15       ,'Maciej Szymczak'              -- pcreator  varchar2
 16      ) result
 17  from dual;

    RESULT
----------
    506861

SQL>
SQL> select api.insert_class(
  2        1
  3       ,date'2010-06-06'               -- pday
  4       ,'1'                            -- phour     number
  5       ,100                            -- pfill     number
  6       ,(select max(id) from subjects) -- psub_id   number
  7      ,(select max(id) from forms)    -- pfor_id   number
  8      ,(select max(id) from lecturers)
  9                                      -- pcalc_lec_ids   varchar2
 10      ,(select max(id) from groups)
 11                                      -- pcalc_gro_ids   varchar2
 12      ,(select max(id) from rooms)
 13                                      -- pcalc_rom_ids   varchar2
 14      ,'BPLATA'                       -- powner    varchar2
 15      ,'Maciej Szymczak'              -- pcreator  varchar2
 16     ) result
 17  from dual;
select api.insert_class(
       *
Bú¹D w linii 1:
ORA-00001: unique constraint (PLANNER.LEC_CLA_U) violated
ORA-06512: at "PLANNER.API", line 84
ORA-06512: at "PLANNER.API", line 151


SQL>
SQL> select api.delete_class(506861) from dual;

API.DELETE_CLASS(506861)
--------------------------------------------------------------------------------



SQL> select api.insert_class(
  2       1
  3      ,date'2010-06-06'               -- pday
  4      ,'1'                            -- phour     number
  5      ,100                            -- pfill     number
  6      ,(select max(id) from subjects) -- psub_id   number
  7      ,(select max(id) from forms)    -- pfor_id   number
  8      ,(select max(id) from lecturers)
  9                                      -- pcalc_lec_ids   varchar2
 10      ,(select max(id) from groups)
 11                                      -- pcalc_gro_ids   varchar2
 12      ,(select max(id) from rooms)
 13                                      -- pcalc_rom_ids   varchar2
 14      ,'BPLATA'                       -- powner    varchar2
```

```
         ,'Maciej Szymczak'              -- pcreator  varchar2
        ) result
     from dual
     ;
    RESULT
----------
    506862
```

DB Table Diagram



```
CREATE TABLE LOOKUPS (
 ID            NUMBER
,TYPE          VARCHAR2(50)
,CODE          VARCHAR2(50)
,NAME          VARCHAR2(50)
,DESCRIPTION   VARCHAR2(50)
);
```

```
CREATE TABLE SYSTEM_PARAMETERS
(
  NAME  VARCHAR2(200)
 ,VALUE VARCHAR2(200)
);
```

```
CREATE TABLE ORG_UNITS (
  ID              NUMBER(10),
  NAME            VARCHAR2(50),
  CODE            VARCHAR2(5),
  PARENT_ID       NUMBER(10),
  STRUCT_CODE     VARCHAR2(500),
  DESC1           VARCHAR2(200),
  DESC2           VARCHAR2(200));
```

```
CREATE TABLE FORM_FORMULAS (
  ID            NUMBER(10)
 ,FOR_ID        NUMBER(10)
 ,ORGUNI_ID     NUMBER(10)
 ,FORMULA_TYPE VARCHAR2(50)
 ,FORMULA       VARCHAR2(254)
 ,DATE_FROM     DATE
 ,DATE_TO       DATE
 ,DESC1         VARCHAR2(200)
 ,DESC2         VARCHAR2(200)
);
```

```
CREATE TABLE FORMS (
  ID               NUMBER(10),
  abbreviation     VARCHAR2(20),
  NAME             VARCHAR2(20),
  DESC1            VARCHAR2(200),
  DESC2            VARCHAR2(200),
  SORT_ORDER_ON_REPORTS);
```

```
CREATE TABLE PLANNERS (
 ID               NUMBER(10),
 NAME             VARCHAR2(30),
 TYPE        VARCHAR2(30));
```

```
CREATE TABLE LEC_PLA (
  ID              NUMBER(10),
  LEC_ID          NUMBER(10),
  PLA_ID          NUMBER(10));
```

```
CREATE TABLE SUB_PLA
```

```
CREATE TABLE LECTURERS (
  ID              NUMBER(10),
  abbreviation    VARCHAR2(20),
  TITLE           VARCHAR2(30),
  FIRST_NAME      VARCHAR2(30),
  LAST_NAME       VARCHAR2(30),
  COLOUR          INTEGER,
  DESC1           VARCHAR2(200),
  DESC2           VARCHAR2(200),
  ORGUNI_ID          NUMBER);
```

```
CREATE TABLE LEC_CLA (
  ID              NUMBER(10),
  LEC_ID          NUMBER(10),
  CLA_ID          NUMBER(10));
```

```
CREATE TABLE CLASSES (
  ID              NUMBER(10),
  DAY             DATE,
  HOUR            NUMBER(2),
  FILL            NUMBER(3),
  SUB_ID          NUMBER(10),
  FOR_ID          NUMBER(10),
  DESC1           VARCHAR2(200),
  DESC2           VARCHAR2(200));
```

```
CREATE TABLE SUBJECTS (
  ID              NUMBER(10),
  abbreviation    VARCHAR2(20),
  NAME            VARCHAR2(30),
  COLOUR          INTEGER,
  DESC1
VARCHAR2(200),
  DESC2
VARCHAR2(200));
```

```
CREATE TABLE GRO_PLA (
  ID              NUMBER(10),
  GRO_ID          NUMBER(10),
  PLA_ID          NUMBER(10));
```

```
CREATE TABLE GROUPS (
  ID                 NUMBER(10),
  abbreviation       VARCHAR2(20),
  NAME               VARCHAR2(30),
  NUMBER_OF_PEOPLES NUMBER(10),
  COLOUR             INTEGER,
  GROUP_TYPE VARCHAR2(50),
  DESC1              VARCHAR2(200),
  DESC2              VARCHAR2(200));
```

```
CREATE TABLE GRO_CLA (
  ID              NUMBER(10),
  GRO_ID          NUMBER(10),
  CLA_ID          NUMBER(10));
```

```
CREATE TABLE ROM_PLA (
  ID              NUMBER(10),
  ROM_ID          NUMBER(10),
  PLA_ID          NUMBER(10));
```

```
CREATE TABLE FOR_PLA
```

```
CREATE TABLE ROOMS (
  ID                 NUMBER(10),
  RESCAT_ID          NUMBER(10),
  NAME               VARCHAR2(20),
  BUILDING           VARCHAR2(30),
  NUMBER_OF_PEOPLES  NUMBER(10),
  COLOUR             INTEGER,
  DESC1              VARCHAR2(200),
  DESC2              VARCHAR2(200)
```

```
CREATE TABLE ROM_CLA (
  ID              NUMBER(10),
  ROM_ID          NUMBER(10),
  CLA_ID          NUMBER(10));
```

```
CREATE TABLE ROL_PLA (
  ID              NUMBER(10),
  ROL_ID          NUMBER(10),
  PLA_ID          NUMBER(10));
```

```
CREATE TABLE PERIODS (
  ID              NUMBER(10),
  NAME            VARCHAR2(50),
  DATE_FROM       DATE,
  DATE_TO         DATE,
  ROL_ID          NUMBER);
```

```
CREATE TABLE RESOURCE_CATEGORIES (
  ID                 NUMBER(10),
  NAME               VARCHAR2(20),
  DESC1              VARCHAR2(200),
  DESC2              VARCHAR2(200));
```

```
CREATE TABLE TIMETABLE (
  ID              NUMBER(10),
  GRO_ID          NUMBER(10),
  PER_ID          NUMBER(10),
  SUB_ID          NUMBER(10),
  FOR_ID          NUMBER(10),
  LIMIT           NUMBER(14,2));
```

```
CREATE TABLE RESERVATIONS (
  ID              NUMBER(10),
  DAY             DATE,
  HOUR            NUMBER(2));
```