Bentley Map

GIS for the Engineering Professional

Creating Oracle Spatial multi-table views

This article (Creating Oracle Spatial Views) discusses creating an Oracle view to limit the number of property columns displayed by Bentley Map. For example, a map reviewer may only need to see 3 or 4 properties while the table may have a dozen or more. Setting up a view that only reveals the needed parcel properties is an ideal solution in this case.

In another situation, you may have a Bentley Map spatial table named PARCELS that's maintained by Bentley Map users and an attribute table named TAXES that's maintained by a data entry clerk. In reviewing workflows, you find that it would be productive if the Bentley Map users had access to a few of the properties in the TAXES table. This would eliminate the data entry clerk from having to correspond with the GIS department in order to update records in the TAXES table.

To set this up, consider the following.

PARCELS table (spatial) is created using GID (Graphic ID) as the primary key. The table also has the following properties: PARCEL_NUM, LAND_TYPE, FIRE_DISTRICT, TOWNSHIP and GEOMETRY.

```
DROP TABLE PARCELS CASCADE CONSTRAINT;

CREATE TABLE "PARCELS"
(
  GID NUMBER CONSTRAINT PARCELS_PK PRIMARY KEY,
  Parcel_Num NUMBER,
  Land_Type VARCHAR2(20),
  Fire_District VARCHAR2(20),
  Township VARCHAR2(20),
  Geometry MDSYS.SDO_GEOMETRY);
```

The spatial metadata is inserted into USER_SDO_GEOM_METADATA:

```
DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='PARCELS';


INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES ('PARCELS', 'Geometry',
    MDSYS.SDO_DIM_ARRAY(
    MDSYS.SDO_DIM_ELEMENT('X', 2197290.78, 2401264.08, 0.00000005),
    MDSYS.SDO_DIM_ELEMENT('Y', 703310.077, 911592.401, 0.00000005)
),
NULL);
```

The spatial index is created:

```
DROP INDEX PARCELS_SIDX;

CREATE INDEX PARCELS_SIDX ON PARCELS(Geometry)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('layer_gtype=POLYGON, sdo_indx_dims=2')
;
```

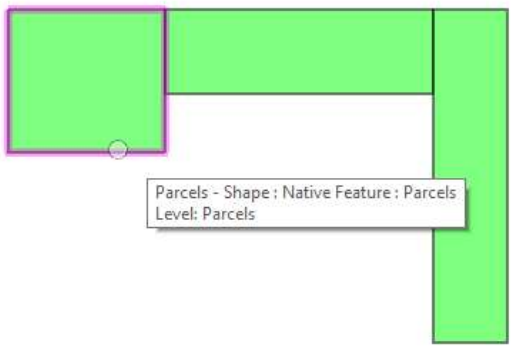
And three records are added to the newly created PARCELS table:

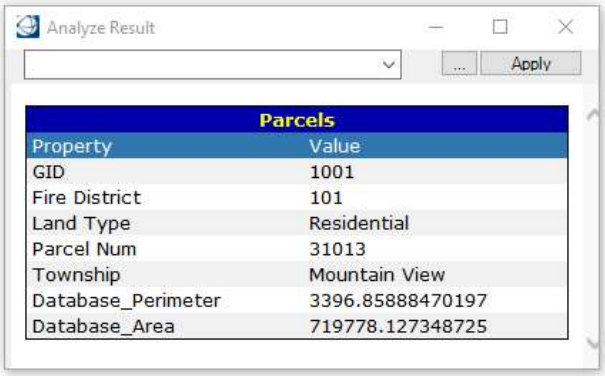
```
Insert into PARCELS (GID,PARCEL_NUM,LAND_TYPE,FIRE_DISTRICT,TOWNSHIP,GEOMETRY) values
(1001,31013,'Residential','101','Mountain View',MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(2298997.85649565,
814652.663198157, 2298997.85649565, 813840.697843586, 2299884.32058343, 813840.697843586,
2299884.32058343, 814652.663198157, 2298997.85649565, 814652.663198157)));



Insert into PARCELS (GID,PARCEL_NUM,LAND_TYPE,FIRE_DISTRICT,TOWNSHIP,GEOMETRY) values
(1002,31014,'Commercial','101','Mountain View',MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(2299884.32058343,
814652.663198157, 2299884.32058343, 814177.9757601, 2301407.54056526, 814177.9757601,
2301407.54056526, 814652.663198157, 2299884.32058343, 814652.663198157)));



Insert into PARCELS (GID,PARCEL_NUM,LAND_TYPE,FIRE_DISTRICT,TOWNSHIP,GEOMETRY) values
(1003,31012,'Industrial','367','Wheatland',MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(2301407.54056526,
814652.663198157, 2301407.54056526, 812766.405220614, 2301832.04449462, 812766.405220614,
2301832.04449462, 814652.663198157, 2301407.54056526, 814652.663198157)));
```

In order to add new records, a sequence is defined. Note that it starts at 1004 since the table has three records which were previously inserted.

```
DROP SEQUENCE "PARCELS_SEQ";

CREATE SEQUENCE PARCELS_SEQ
 START WITH 1004 INCREMENT BY 1;
```

At this point, the PARCELS table could be registered as a spatial feature and be used in a Bentley Map project. Note that Bentley Map is only seeing the properties from the PARCELS table.





The remainder of this article discusses how to set up the Parcel feature to also see properties from the TAXES table.

The TAXES table (non-spatial) is created with the following properties: AID (Attribute ID), PARCEL_NUM, TAX_DISTRICT, TAX_VALUE.

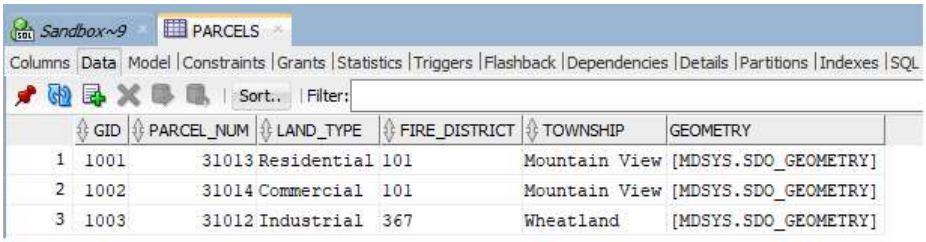
```
DROP TABLE TAXES CASCADE CONSTRAINT;

CREATE TABLE "TAXES"
 (
AID NUMBER PRIMARY KEY,
Parcel_Num NUMBER,
Tax_District VARCHAR2(10),
Tax_Value NUMBER);
```

Three records are inserted to correspond with the PARCELS table. Note that the ID number and PARCEL_NUM match in this case.

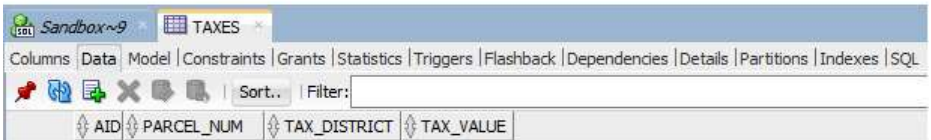
```
INSERT INTO TAXES (AID, Parcel_Num, Tax_District, Tax_Value)
VALUES (1001, '31013', 'MV-850', '6500');

INSERT INTO TAXES (AID, Parcel_Num, Tax_District, Tax_Value)
VALUES (1002, '31014', 'MV-850', '6375');

INSERT INTO TAXES (AID, Parcel_Num, Tax_District, Tax_Value)
VALUES (1003, '31015', 'WL-050', '10185');
```

Since the TAXES table is non-spatial and is only used to store attribute information, it does not need to be added to USER_SDO_GEOM_METADATA and a spatial index doesn't need to be created.

Reviewing the tables in Oracle shows the following:



	GID	PARCEL_NUM	LAND_TYPE	FIRE_DISTRICT	TOWNSHIP	GEOMETRY
1	1001	31013	Residential	101	Mountain View	[MDSYS.SDO_GEOMETRY]
2	1002	31014	Commercial	101	Mountain View	[MDSYS.SDO_GEOMETRY]
3	1003	31012	Industrial	367	Wheatland	[MDSYS.SDO_GEOMETRY]



	AID	PARCEL_NUM	TAX_DISTRICT	TAX_VALUE
1	1001	31013	MV-850	6500
2	1002	31014	MV-850	6375
3	1003	31015	WL-050	10185

The next step is to create the multi-table view. In this example, all of the columns of the PARCELS table are revealed, but only the TAX_DISTRICT column from the TAXES table is included. This will allow the Map user to update all properties in the view regardless of the table they reside in.

```
DROP VIEW PARCEL_TAX_VIEW;

CREATE OR REPLACE VIEW PARCEL_TAX_VIEW AS
SELECT PARCELS.GID, PARCELS.PARCEL_NUM, PARCELS.LAND_TYPE, PARCELS.FIRE_DISTRICT,
PARCELS.TOWNSHIP, PARCELS.GEOMETRY, TAXES.TAX_DISTRICT
FROM PARCELS, TAXES
WHERE  PARCELS.GID = TAXES.AID;
```

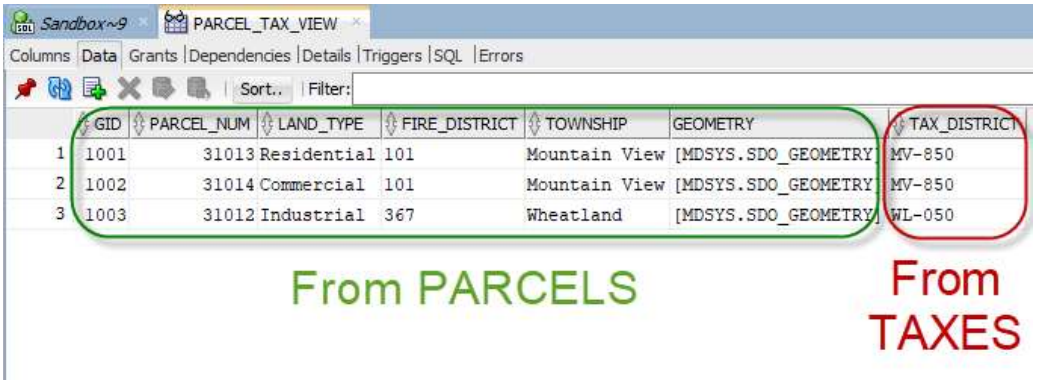
Bentley Map requires a primary key be defined on the spatial table. However you can't have a primary key on a view. To get around this limitation, a primary key is created and then disabled.

```
ALTER VIEW PARCEL_TAX_VIEW ADD PRIMARY KEY (GID) DISABLE;
```

Since the view will be registered as a spatial feature, it will be added to USER_SDO_GEOM_METADATA:

```
DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='PARCEL_TAX_VIEW';

INSERT INTO USER_SDO_GEOM_METADATA
 SELECT 'PARCEL_TAX_VIEW', COLUMN_NAME, DIMINFO, SRID
 FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='PARCELS'
```

Taking a look at the view in Oracle shows the following:

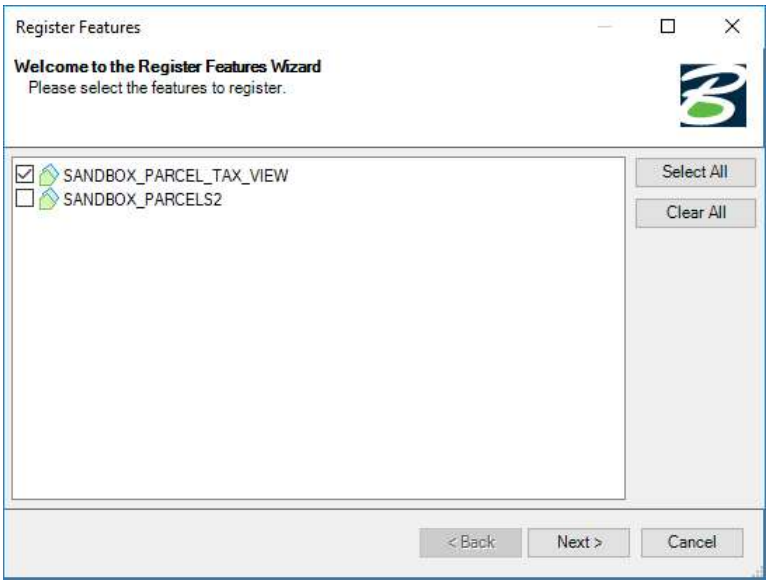


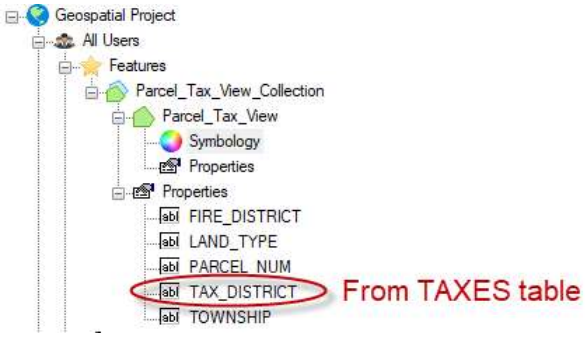
To accommodate INSERT, UPDATE and DELETE statements, Oracle enables you to define procedures that are implicitly executed against associated tables. These procedures are called database triggers and are defined as follows. These triggers are required to update the correct tables when new parcels are added, existing parcels are edited or deleted.

```
DROP TRIGGER PARCEL_TAX_VIEW_INS;

CREATE OR REPLACE TRIGGER PARCEL_TAX_VIEW_INS
INSTEAD OF INSERT ON PARCEL_TAX_VIEW
BEGIN
  INSERT INTO PARCELS
   VALUES (:new.GID, :new.PARCEL_NUM, :new.LAND_TYPE, :new.FIRE_DISTRICT, :new.TOWNSHIP,
:new.GEOMETRY);

  INSERT INTO TAXES (AID, PARCEL_NUM, TAX_DISTRICT) VALUES (:new.GID, :new.PARCEL_NUM,
:new.TAX_DISTRICT);

END PARCEL_TAX_VIEW_INS;
/



DROP TRIGGER PARCEL_TAX_VIEW_UPD;

CREATE OR REPLACE TRIGGER PARCEL_TAX_VIEW_UPD
INSTEAD OF UPDATE ON PARCEL_TAX_VIEW
BEGIN
  UPDATE PARCELS SET
   GID = :new.GID, PARCEL_NUM = :new.PARCEL_NUM, LAND_TYPE =  :new.LAND_TYPE,
FIRE_DISTRICT = :new.FIRE_DISTRICT, TOWNSHIP = :new.TOWNSHIP, GEOMETRY = :new.GEOMETRY
  WHERE GID = :old.GID;

  UPDATE TAXES SET
  AID = :new.GID, PARCEL_NUM = :new.PARCEL_NUM, TAX_DISTRICT = :new.TAX_DISTRICT
  WHERE AID = :old.GID;

END PARCEL_TAX_VIEW_UPD;
/
```

```
DROP TRIGGER PARCEL_TAX_VIEW_DEL;

CREATE OR REPLACE TRIGGER PARCEL_TAX_VIEW_DEL
  INSTEAD OF DELETE ON PARCEL_TAX_VIEW
BEGIN
 DELETE FROM PARCELS WHERE GID = :old.GID;
 DELETE FROM TAXES WHERE AID = :old.GID;
END PARCEL_TAX_VIEW_DEL;
/
```

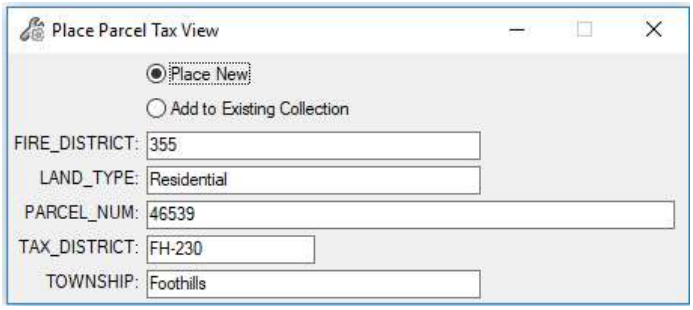
At this point, the view can be registered as a spatial feature in the Geospatial Administrator.



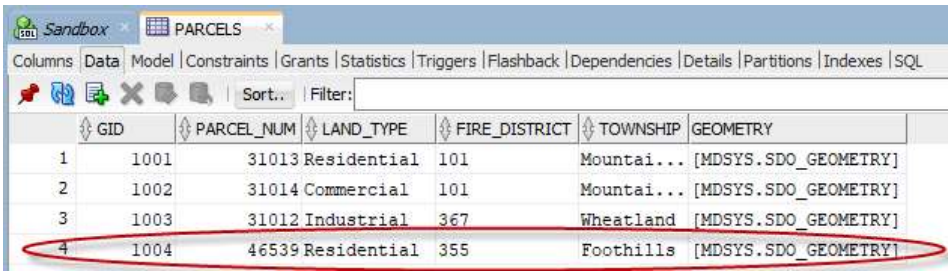


Adding new parcels updates both the PARCELS and TAXES table.

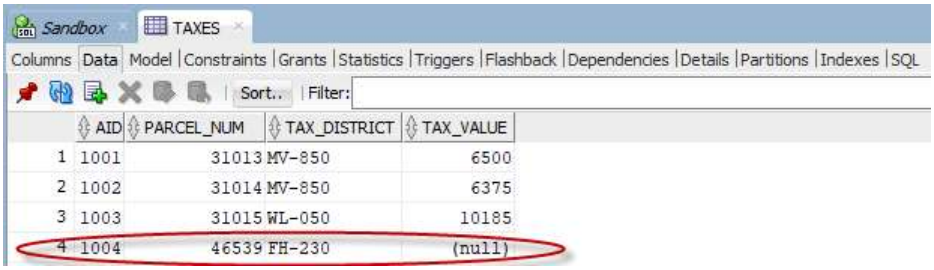
For example, adding the following new parcel:



...inserts the fourth row into the PARCELS and TAXES tables. Notice that the TAX_VALUE property remains NULL since the Map user did not have access to that property.



	GID	PARCEL_NUM	LAND_TYPE	FIRE_DISTRICT	TOWNSHIP	GEOMETRY
1	1001	31013	Residential	101	Mountai...	[MDSYS.SDO_GEOMETRY]
2	1002	31014	Commercial	101	Mountai...	[MDSYS.SDO_GEOMETRY]
3	1003	31012	Industrial	367	Wheatland	[MDSYS.SDO_GEOMETRY]
4	1004	46539	Residential	355	Foothills	[MDSYS.SDO_GEOMETRY]



	AID	PARCEL_NUM	TAX_DISTRICT	TAX_VALUE
1	1001	31013	MV-850	6500
2	1002	31014	MV-850	6375
3	1003	31015	WL-050	10185
4	1004	46539	FH-230	(null)

Similarly, editing or deleting existing parcels will update the tables accordingly.

The attached ZIP file contains:

A PDF of this article for printing.

The SQL script for PARCELS - TAXES

And a generic script for TABLE1 - TABLE2 that you can use as a starting point.

These scripts have been tested on Oracle 12c with Bentley Map V8i and Bentley Map CONNECT Edition.

These scripts are supplied as-is and do not come with support. Bentley Systems Inc., and the author of these scripts assume no liability for damages direct, indirect, or consequential, which may result from the use of these scripts. Use these scripts at your own risk.

As always, please ensure you're doing all experimenting in a test database.