May 2013
Oracle Spatial and Graph
User Conference

ORACLE

May 22, 2013

Ronald Reagan Building and International Trade Center

Washington, DC USA

Glenn Kronschnabl

VP, Software Engineering &
Technology, CoreLogic, Inc.

CoreLogic
Detection of Parcel Change (DPC) Process

CoreLogic®

OVERVIEW OF DPC

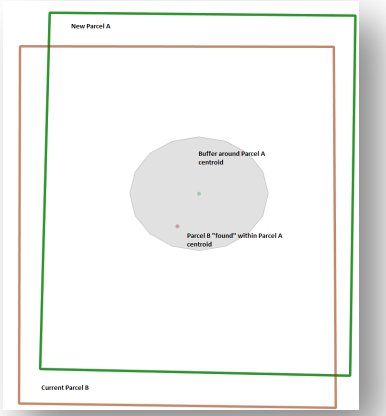
- The identification of changes to property parcel characteristics in response to everyday market transactions.
- Identified changes are validated against a previous state in our system of record to identify whether a parcel continues to exist as is, or whether modifications have occurred.

OPPORTUNITIES / CHALLENGES

- **Opportunity**: DPC is a robust processing model that tracks spatial and temporal changes, providing a rich dataset equipped to answer a variety of new property-related questions.
- **Challenge**: The varying definitions for “property” which can be defined by the Assessor ‘s Parcel Number (APN) or the physical address. The Tax ID is yet another identifier.

SOLUTIONS

- CoreLogic ParcelPoint® defines a property as a parcel geometry or shape.
- CoreLogic matches “geometries” within the ParcelPoint loading process.
- Textual and numerical attributes are **not** part of the ParcelPoint unique ID matching mechanism.



RESULTS

- Changes to parcels are registered in the ParcelPoint database, and are also assigned a unique ID.
- A register is used to keep a view to the lineage of the parcel through time.
- New parcels are constantly being created.
- CoreLogic continually incorporates new parcels into the database, allowing us to identify parcels where new development has occurred.

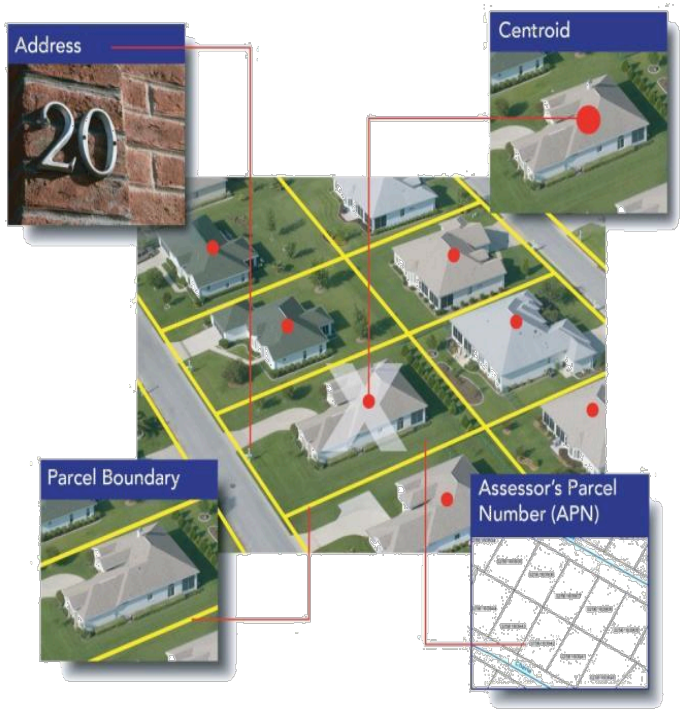
Detecting Property Change in a Spatial World: A Unique Parcel ID Implementation

Program Agenda

- Parcel Data Defined
- Parcels as the Relational Link
- Geocoding Levels of Accuracy & Granularity
- Differentiators in Location Information
- “Detection of Parcel Change” Process
- PolyMatch, MIS & SIG

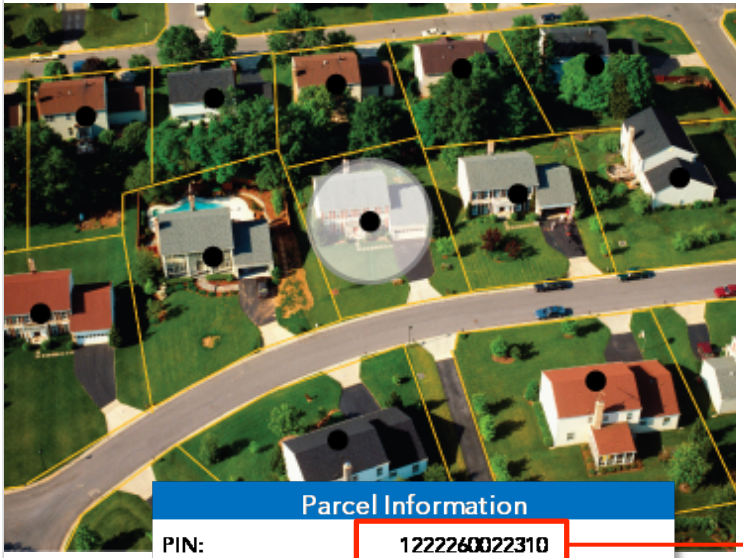
What is Parcel Data?

- Parcel boundary data represents the legal extents of each taxable U.S. property address
- There are an estimated 144.3 million privately owned parcels in the U.S.
- CoreLogic has converted and normalized 135 million parcels from over 2000+ state, county, city, and town sources
- Advanced LBS applications and analytics are starting to exploit the parcel boundary geometry to enhance:
  - Geocoding accuracy
  - Risk assessment
  - Risk concentration
  - Other uses where “granular” accuracy is important



Parcels as the Relational Link

Geocode	
Latitude	25.898951
Longitude	-80.126806
Address Line	276 BAL BAY DR
City/State Zip	MIAMI BEACH FL 33154
PxPoint Data Set	PARCEL
Elevation, Slope, and Aspect	
Elevation (Feet)	1.31
Slope (Degrees)	0
Aspect	Flat
Mainland Determination & Distance	
Distance to Seaward Water Feature	101 feet
Seaward Water Feature Name	Biscayne Bay
Mainland: Yes or No	No
Coastal Storm Surge	
Risk Value	5
Risk Level	Extreme
Hurricane Landfall Probability	
% Tropical Storm Risk (Winds 39 - 73mph)	5.3
% Tropical Storm Risk (50-yr)	93.5
% Hurricane Risk (Cat 1-5 Storms)	1.6
% Hurricane Risk (50-yr)	56.3
% Intense Hurricane Risk (Cat 3-5 Storms)	0.4
% Intense Hurr. Risk (50-yr)	19.9
Flood Risk	
Flood Hazard Zone	AE
Undeveloped Coastal Barrier Area	COBRA_OUT
Special Flood Hazard Area (SFHA)	IN
Damaging Winds	
Straight Line Wind (SLW) Risk	Moderate
SLW Frequency	1 Event Every 4 - 6 Years
Hurricane Risk	Very High
Hurricane Frequency	1 Event Every 3 - 5 Years
Tornado Risk	Moderate
Tornado Frequency	1 Event every 5 - 8 Years
Sinkhole	
Risk	Low
Distance to Very High Sinkhole Risk	Greater than 10 miles
Wildfire Risk	
Brushfire Risk	Urban
Nearest high-risk value	Very High
Distance to High/Very High	> 1 mile



Parcel Information

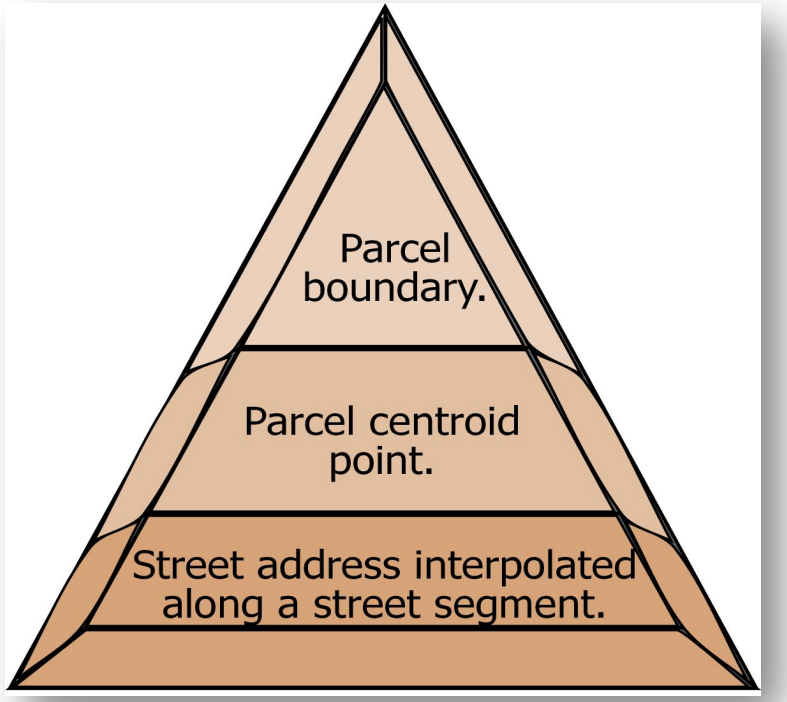
PIN:	1222260022310
Address Line:	276 BAL BAY DR
City/ State/ Zip:	BAL HARBOUR FL 33154
Latitude:	25.898951
Longitude:	-80.126806

PIN:	1222260022310
Property Address:	276 BAL BAY DR
Owner:	BEV SIEVERT
Land Value:	$9,892,934
Building Value:	$2,349,327
Market Value:	$12,242,261
Assessed Value:	$9,375,066
Adj Sq Footage:	9,988
Year Built:	1977
Bedrooms:	9
Baths:	10
Stories:	2
Living Units: 2	2
Adj Sq Footage:	9,988
Lot Size (Sq Ft):	46,279
Year Built:	1977
Construction:	Composite
Pool:	In Ground
Roof Cover:	Tile

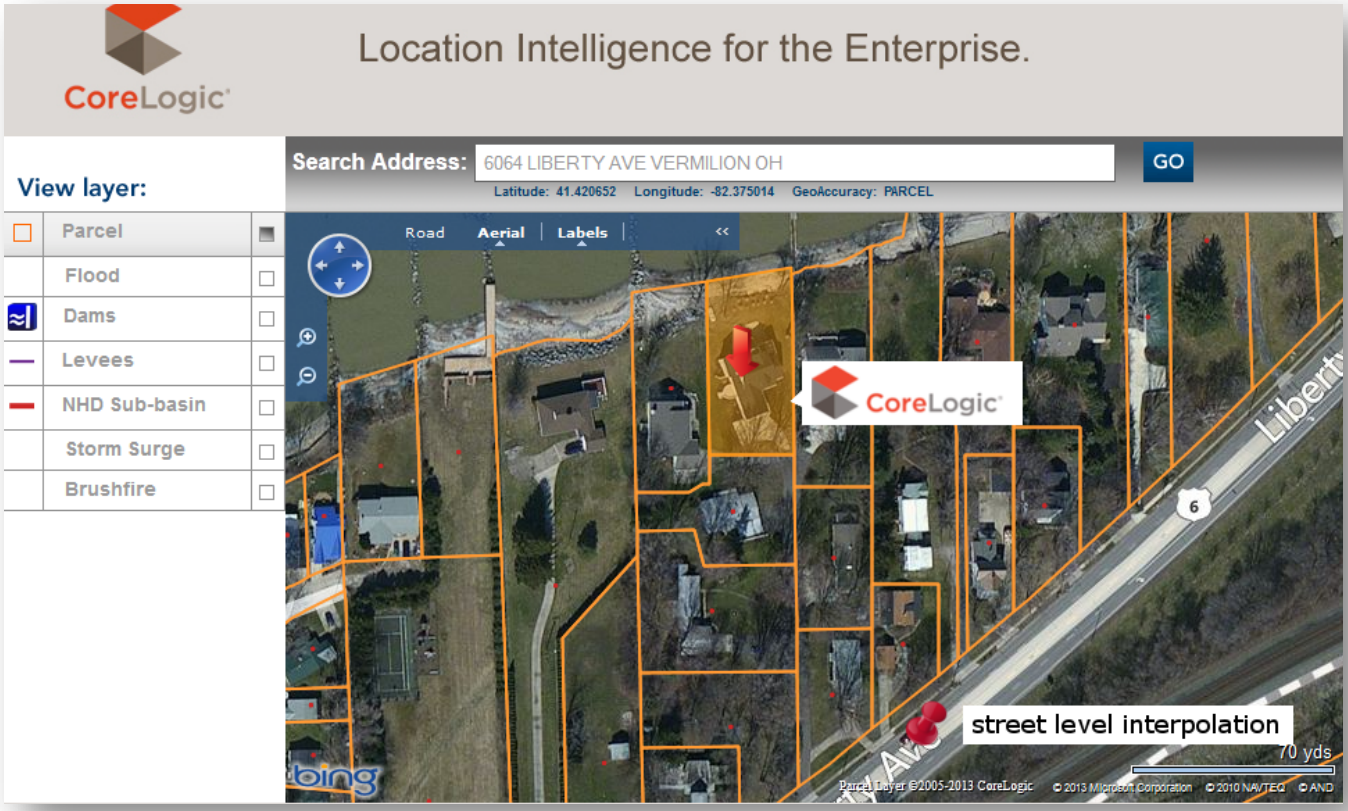
Accurate Hazard Assignment Requires an High Accurate Parcel Level Geocoder

- Location intelligence has evolved to the most “granular” level possible with the availability of digital property parcel boundaries
- Companies can evaluate a variety of natural hazard risks such as coastal storm surge and wildfire using parcel data and parcel level geocoders
- Parcel data allows companies to understand hazard risk and risk concentration at the “micro-level” resulting in:
  - Evaluation of risk at the insured boundary level
  - Heightened profitability
  - Improved business decisioning

Geocoding Levels of Accuracy

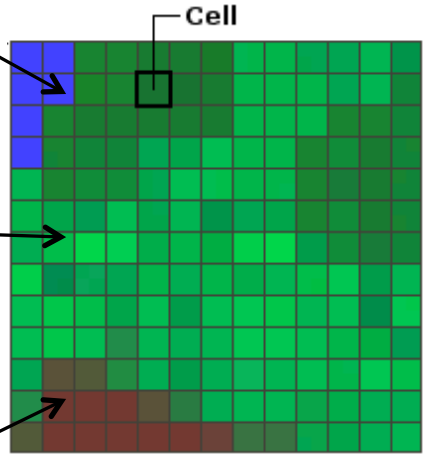


Parcel Level vs. Street Level Geocoding



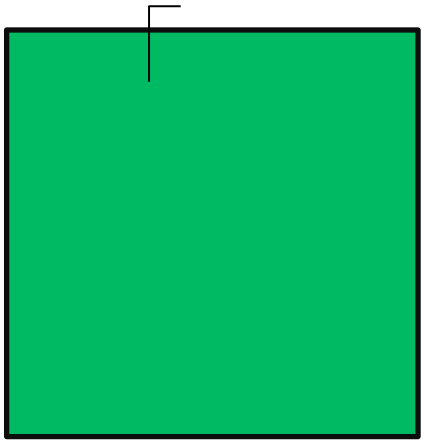
Granularity: Conceptual Illustration

Smaller, More Granular
Area Cell
30 Meter X 30 Meter



1100 (30M X 30M) Cells

Larger, Coarser
Area Cell
1 Kilometer X 1 Kilometer



1 (1 Km Cell)

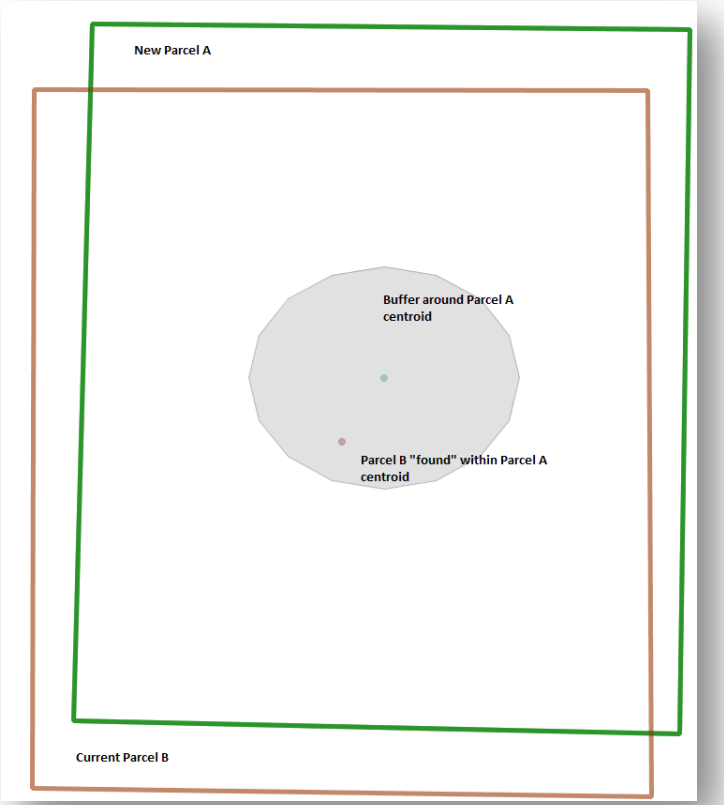
Differentiators in Location Information

- Currency and accuracy are two critical differentiators
- To achieve relevant currency, source data is continuously re-normalized and re-loaded into the database
- **The problem:** Maintaining a meaningful parcel lineage
- **The solution**: Use the parcel geometry as the unique identifier

CoreLogic has developed an in-database “Detection of Parcel Change” (DPC) process to identify and account for changes to the geometry over time by leveraging Oracle Spatial technology

Detection of Parcel Change (DPC) Process

- DPC v1.0 used to identify parcels in a new dataset against those already in our database
- By deriving and matching the geometrical centroid for each parcel against existing centroids, DPC can make a reliable assumption on whether a parcel already exists
- The assumption: If the buffer around the centroid of the new parcel finds a single centroid in the database, we have a match



A Closer Look at the DPC Process

- Process is based on years of knowledge acquiring, processing and loading parcels
- We can identify, in a “wholesale” fashion, parcel loads that might require further review
- Parcel loads should show 85% consistency from update to update
- Our automated loading process audits data elements to alert users that a dataset might need additional review (i.e. if a data threshold is exceeded)

Example: Automated Notification for Parcel Load

- 4,794 parcels reviewed by DPC against 5,978 parcels
- Scheduler deemed this a DPC issue and set the load to **Review** status
- The user can check the validity of the DPC outcome
- Must be reviewed for accuracy

Scheduler v2.6

*** AUDIT Results ***
Status: REVIEW

Issues
... 49 % DPC Unmatched - Exceeds Threshold

Name: 16063_load.shp **Date:** 01/25/2013 17:55:56 **Prj:** GCS_WGS_1984 **Geom:** MULTIPOLYGON
Fields: the_geom.STATE_CODE.CNTY_CODE.REVID.APN.APN2.APN3.TAX_ID.OWNER.
ADDR.CITY.STATE.ZIP.PLUS.GIS_TMP.DPC_ADDR.DPC_APN.DPC_OWNER.DPC_MISC.
ORIG_FID.EMPID.DUPE.TRUE_DUPE.ADJUSTED.VOIDS.PNTS.P_AREA.RATIO.FIP.
NSSDA.LINKER.
County(ies): LINCOLN, ID - 16063.
Rev(s): 25645.
User(s): 3105.
Invalid Geom #: 0

Stats:

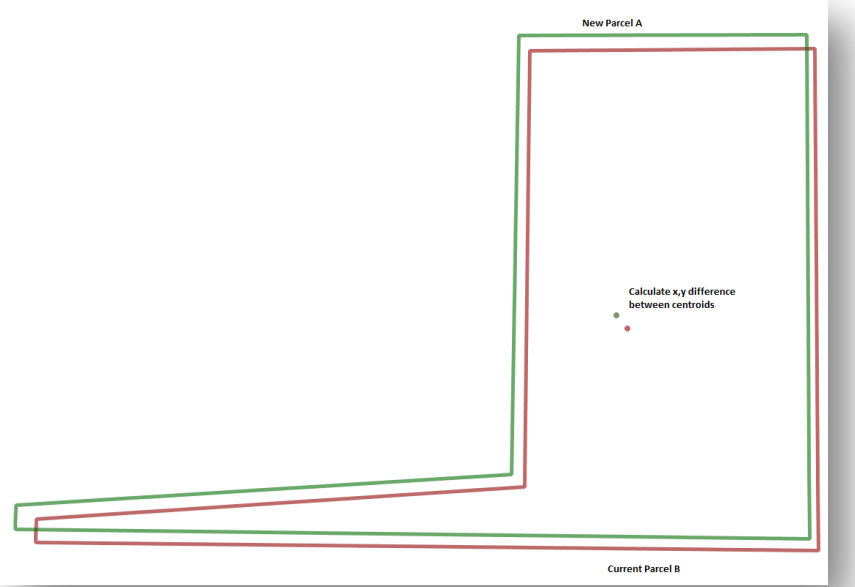
	Revs	Total	Addr	Owner	APN	APN3
SHAPEFILE	1	4,794	1,960	3,574	4,794	0
LOADED	1	5,978	1,772	0	4,498	0
Δ (+/-)	0	-1,184	188	3,574	296	0
Δ (%)		-20	10	100	6	0
DPC	UnMatched	Multihit	Matched	Δ Addr	Δ Owner	Δ APN
#	2,337	6	2,451	38	0	67
%	49	0	51	2	0	3

DPC v2.0

- DPC v1.0 has been in use for three years
- The next generation, DPC v2.0 is in development
- With v2.0 we will only identify parcels based on shape or geometry
- The match will be by “geometry” and not just parcel attributes such as address and APN
- V2.0 will use Oracle Spatial processing technology to identify a relationship **between** any two geometries in space

PolyMatch

- A PolyMatch begins when an input geometry has been deemed a “prospect match”
- Takes the input geometry and prospect and determines the spatial relationship between them:
  - Calculates the X and Y differences
  - Moves the input geometry on top of the prospect
  - Asserts whether the two geometries are the same
- An **EXACT** match implies the same footprint (i.e. number of nodes and place in space)



These geometries are not the same so PolyMatch will not report an **EXACT** match

PolyMatch API

```
-- Performs SDO_RELATE between two geometries
-- @param this_geom
-- @param that_geom
-- @return Intersection pattern. i.e. 'EQUAL', 'INSIDE', 'CONTAINS', etc ..
FUNCTION GET_INTERSECTION_PATTERN(this_geom IN SDO_GEOMETRY, that_geom IN SDO_GEOMETRY) RETURN VARCHAR2;

-- Performs SDO_RELATE between two geometries well known text
-- @param this_geom
-- @param that_geom
-- @return Intersection pattern. i.e. 'EQUAL', 'INSIDE', 'CONTAINS', etc ..
FUNCTION GET_INTERSECTION_PATTERN(this_geom_wkt IN CLOB, that_geom_wkt IN CLOB) RETURN VARCHAR2;

-- Compares spatial relationship between two geometries based on an optional level comparison rule.
-- @param this_geom
-- @param that_geom
-- @param comparison_type
-- @return PolyMatch info object with TRUE or FALSE outcome and the values, if applied, for feature correlation Distance and Symmetric Difference Area
FUNCTION COMPARE(this_geom IN SDO_GEOMETRY, that_geom IN SDO_GEOMETRY, comparison_type IN VARCHAR2 DEFAULT 'STRICT') RETURN POLYMATCH_INFO;

-- Derives the Feature Correlation Distance between two geometries
-- @param first_geom
-- @param second_geom
-- @return The feature correlation distance in meters between both geometries
FUNCTION GET_FEATURE_CORRELATION_DIST(first_geom IN SDO_GEOMETRY, second_geom IN SDO_GEOMETRY) RETURN NUMBER;

-- Derives the Symmetric Difference Area between two geometries
-- @param first_geom
-- @param second_geom
-- @return The Symmetric Difference Area in Square Meters between both geometries
FUNCTION GET_SYMMETRIC_DIFFERENCE_AREA(first_geom IN SDO_GEOMETRY, second_geom IN SDO_GEOMETRY) RETURN NUMBER;

-- Shifts the position of a geometry by x and y changes applied to its centroid
-- @param the_geom
-- @param x
-- @param y
-- @return The geometry shifted position based on its centroid relocation driven by values x and y
FUNCTION SHIFT(the_geom IN SDO_GEOMETRY, x IN NUMBER, y IN NUMBER) RETURN SDO_GEOMETRY;
```

- Intersection pattern
- Compare SDO geometry
- Get feature correlations distance
- Get symmetric difference area
- Shift SDO geometry

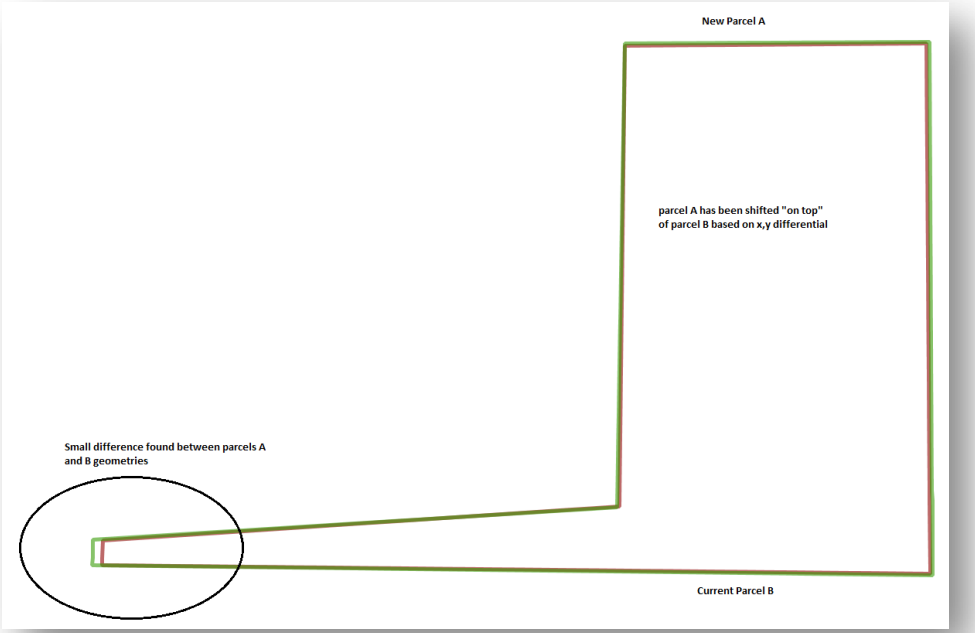
An “EQUAL” Match

```
dbms_output.put_line('==> compare(this_geom, that_geom, comparison_type => '''||comparison_type||''')');

-- get first geometry centroid and vertices
-- this is the main geometry
this_geom_centroid := SDO_GEOM.SDO_CENTROID(this_geom, 0.05);
SELECT v.x, v.y
  INTO this_geom_x, this_geom_y
  FROM dual, TABLE(SDO_UTIL.GETVERTICES(this_geom_centroid)) v;

dbms_output.put_line('this_geom_x: '||this_geom_x);
dbms_output.put_line('this_geom_y: '||this_geom_y);

-- get second geometry centroid and vertices
that_geom_centroid := SDO_GEOM.SDO_CENTROID(that_geom, 0.05);
SELECT v.x, v.y
  INTO that_geom_x, that_geom_y
  FROM dual, TABLE(SDO_UTIL.GETVERTICES(that_geom_centroid)) v;

dbms_output.put_line('that_geom_x: '||that_geom_x);
dbms_output.put_line('that_geom_y: '||that_geom_y);

-- obtain the x and y diff - inverse since we're moving towards first geom
x_diff := that_geom_x - this_geom_x;
y_diff := that_geom_y - this_geom_y;

-- shift the this_geom
this_geom_shifted := SHIFT(this_geom, x_diff, y_diff);
-- validate that they are equal
IF GET_INTERSECTION_PATTERN(this_geom_shifted, that_geom) = 'EQUAL' THEN

    -- straight up match. Only value returned is the result
    RETURN POLYMATCH_INFO('TRUE', NULL, NULL, NULL);

END IF;
```

```
-- mainly a wrapper for simplifying calls to shift a geometry
-- using translation from sdo_util.affinetranforms

dbms_output.put_line('==> shift('||x||', '||y||')');

SELECT SDO_UTIL.AFFINETRANSFORMS(
    geometry => the_geom,
    translation => 'TRUE',
    tx => x,
    ty => y,
    tz => 0.0,
    scaling => 'FALSE',
    pscl => NULL,
    sx => 0.0,
    sy => 0.0,
    sz => 0.0,
    rotation => 'FALSE',
    pl => NULL,
    linel => NULL,
```

If Not an Exact Match, Then What?

- PolyMatch tries other comparisons by deriving the **maximum distance of one geometry to the nearest point in the other geometry**.
- Values are matched against thresholds for a “Tolerance Match”
- If the tolerance is not met, the outcome will be **FALSE**
- PolyMatch looks for the initially-unmatched geometry using a buffer
- Buffer set around the unmatched geometry and neighboring geometries in the buffer



Further Analysis

- Feature Correlation Distance = Maximum difference between geometries
- Symmetric Difference Area = SDO_GEOM.SDO_XOR

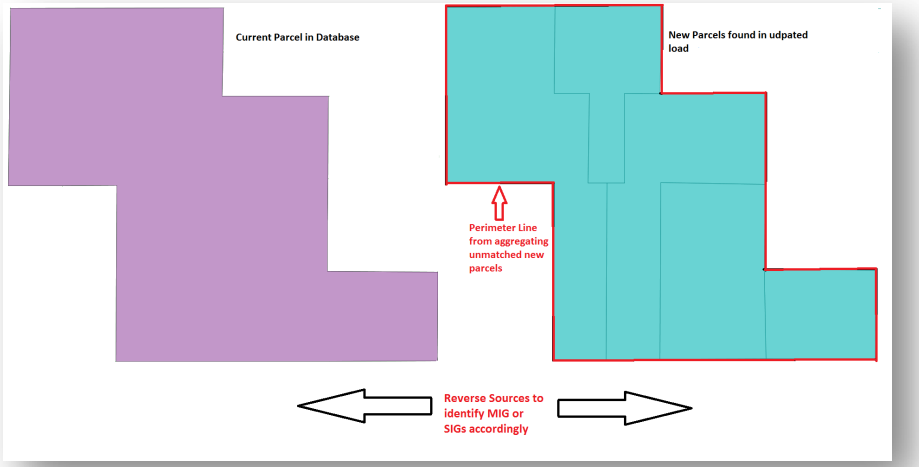
```
-- see if it can be considered a match based on feature correlation distance and symmetric difference
fc_dist := GET_FEATURE_CORRELATION_DIST(this_geom_shifted, that_geom_clone);    -- feature correlation distance call
dbms_output.put_line('fc_dist: '||fc_dist);
-- use perimeter distance as a function of size against fc dist
fc_dist_pct := ROUND((fc_dist * 100) / SDO_GEOM.SDO_LENGTH(this_geom_shifted, 0.05), 2);
dbms_output.put_line('fc_dist_pct: '||fc_dist_pct);

sd_area := GET_SYMMETRIC_DIFFERENCE_AREA(this_geom_shifted, that_geom_clone);  -- symmetric difference area call
dbms_output.put_line('sd_area: '||sd_area);

this_geom_shifted_area := ROUND(SDO_GEOM.SDO_AREA(this_geom_shifted, .05, 'UNIT=SQ_M'), 2);
sd_area_pct := ROUND((sd_area * 100) / this_geom_shifted_area, 2);
dbms_output.put_line('sd_area_pct: '||sd_area_pct);
```

Outcome: Match or Buffer Exhausted

- PolyMatch performed until there's a match or the buffer is exhausted
- The Parcel Load System monitors and tags a unique identifier for each new parcel
- New parcels are:
  - Tagged with a new ID
  - Checked for attribute changes
- Those not found will be stored

Inferred Geometries: Merges & Splits

- Inferring Geometry Change: Detects what might have happened to a parcel geometry
- Infer outcome based on analysis and data updates
- New data and unmatched records process through the database
- Match new record geometry against unmatched records to check for a Merged Inferred Geometry (MIG)
- Reverse sources and use the same method to look for new records that split from earlier records through Split Inferred Geometry (SIG)



Example: A large parcel splits into smaller parcels for subdivision land

Oracle Environment at a Glance

- **Primary**:
  - OS: SUSE Linux Enterprise Server 10 (x86_64)
  - VERSION = 10
  - PATCHLEVEL = 4
- **Hardware**:
  - 32core xeon 64GB
- **Oracle**:
  - Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
- **Parcel configuration**:
  - 1.7 TB of data files allotted
- **Options**:
  - Active Data Guard
  - Partitioning
- **Data highlights**:
  - 135M parcels
  - Approx. 2B nodes

Oracle Spatial Functions Used:

- SDO_RELATE
- SDO_GEOM.SDO_AREA
- SDO_GEOM.SDO_XOR
- SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
- SDO_GEOM.SDO_BUFFER
- SDO_GEOM.SDO_DISTANCE
- SDO_UTIL.POLYGONTOLINE
- SDO_UTIL.GETNUMVERTICES
- SDO_UTIL.SIMPLIFY
- SDO_UTIL.AFFINETRANSFORMS

Summary: Why DPC Matters to CoreLogic

- Parcel currency is a ParcelPoint® differentiator that requires continuous updating and loading of new/updated parcel data
- DPC v2.0 is a process allowing identification of changes to property parcel characteristics in response to everyday market transactions
- DPC v2.0 allows CoreLogic to manage the continually changing parcel fabric in an efficient in-database manner
- DPC v2.0 is a robust processing model that tracks spatial and temporal changes, providing a rich dataset equipped to answer a variety of new property-related questions

Q&A