문제점
상황. 접속 할때 생성되는 session 정보가 유저당 고유의 토큰을 부여 한다.

1. 하루의 한 유저가 아침에 접속하고 저녁에 접속하면 두번 접속 한 것이지만 유저당 한 토큰 값을 부여 하기 때문에 구별 할 수가 없다.
 - 접속 시간을 기준으로 1시간을 초과하면 다른 접속 시간으로 간주 한다.

2. xnuvo.com접속 한 위치가 상당수 많다. 그 이유는 사이트 내부에서 쇼핑을 하면서 여러 로그를 남기기 때문이다. 따라서 처음에 외부 접속 시점에 referer 값을 xnuvo.com을 대체 해야 한다.
 하지만 한 유저당 토큰 값이 같으므로 처음에 접속한 referer 값을 구별 할 수 있는 방법이 없다.

3. 접속된 유저 정보가 순차적으로 정렬 안되는듯 싶다.

```
20121116171928          -       /x/users/sign_in?path=%2Fyanemochi%3Flocale%3Dko-KR
20121116171929          -       /
20121116171930  xnuvo.com       http://xnuvo.com/       /
20121116171937  xnuvo.com       http://xnuvo.com/       /x/periodicals?u=3095
20121116171941          -       /favicon.ico
20121116171941  daum.net        http://widgetprovider.daum.net/gadgets/ifr?container=default&widgetId=224&mid=0&nocache=0&c
ountry=ALL&lang=ALL&view=default&parent=http%3A%2F%2Fblog.naverblogwidget.com&st=null%3Anull%3Anull%3Anull%3Ahttp%253A//wid
getcfs1.daum.net/xml/14/widget/2009/01/05/18/01/4961cc789ac24.xml%3A224%3Anull&url=http%3A%2F%2Fwidgetcfs1.daum.net%2Fxml%2
F14%2Fwidget%2F2009%2F01%2F05%2F18%2F01%2F4961cc789ac24.xml     /yanemochi
20121116171942  daum.net        http://widgetprovider.daum.net/gadgets/ifr?container=default&widgetId=224&mid=0&nocache=0&c
ountry=ALL&lang=ALL&view=default&parent=http%3A%2F%2Fblog.naverblogwidget.com&st=null%3Anull%3Anull%3Anull%3Ahttp%253A//wid
getcfs1.daum.net/xml/14/widget/2009/01/05/18/01/4961cc789ac24.xml%3A224%3Anull&url=http%3A%2F%2Fwidgetcfs1.daum.net%2Fxml%2
F14%2Fwidget%2F2009%2F01%2F05%2F18%2F01%2F4961cc789ac24.xml     /yanemochi?locale=ko-KR
20121116171943  xnuvo.com       http://xnuvo.com/yanemochi?locale=ko-KR /yanemochi?locale=ko-KR
20121116171944  xnuvo.com       http://xnuvo.com/yanemochi?locale=ko-KR /yanemochi?locale=ko-KR
20121116171945  xnuvo.com       http://xnuvo.com/       /x/periodicals?u=3095
20121116171948  xnuvo.com       http://xnuvo.com/yanemochi?locale=ko-KR /yanemochi/categories/835?ref=categories%2Findex
20121116171951  xnuvo.com       http://xnuvo.com/yanemochi?locale=ko-KR#!/yanemochi/categories/835      /x/periodicals?u=30
95
20121116171954  xnuvo.com       http://xnuvo.com/       /x/periodicals?u=3095
```

1. HIVE Specification

로그를 하둡 에서 분석 할 때 사용되는 로그 데이터 구조를 정의 한다.

delimeter	\t
referer	http://www.facebook.com → facebook.com

mysql import	
EXPORT TO CSV	SELECT * INTO OUTFILE s.CSV’ FIELDS TERMINATED BY ‘,’ OPTIONALLY ENCLOSED BY”’ LINES TERMINATED BY ‘\n’ FROM YOUR_TABLE;
INFLOW DATA EXPORT TO MYSQL	cat 0* > data mysql -uroot -p --local-infile=true use honeybee; LOAD DATA LOCAL INFILE '/root/honeybee/inflow/merged' INTO TABLE honeybee_graph_1_data (time, referer, count, female, male); DELETE FROM honeybee_graph_1_data WHERE referer = ''; UPDATE honeybee_graph_1_data SET female = 0 WHERE female is null; UPDATE honeybee_graph_1_data SET male = 0 WHERE male is null;

setting command	add jar /usr/local/src/hive-0.9.0/lib/hive-contrib-0.9.0.jar;
	add jar /usr/local/hive/hive_aux/honeybee-hadoop.jar;
	add file /usr/local/hive/hive_aux/honeybee-hadoop.jar;
	create temporary function dateToHour as
	'org.honeybee.hive.DateToHour';
	create temporary function dateToDay as
	'org.honeybee.hive.DateToDay';
	create temporary function dateToMin as
	'org.honeybee.hive.DateToMin';
	create temporary function dateToSec as
	'org.honeybee.hive.DateToSec';
	create temporary function simpleReferer as
	'org.honeybee.hive.SimpleReferer';
	create temporary function lifetime as 'org.honeybee.hive.LifeTime';
	create temporary function logRegex as 'org.honeybee.hive.LogRegex'
	
	<!-- 더이상 사용하지 않음 -->
	set mapred.max.split.size=1000000;
	set hive.optimize.bucketmapjoin=true;
	set hive.optimize.bucketmapjoin.sortedmerge=true;
	set hive.groupby.skewindata=true;
	set hive.merge.size.per.task=256000000;
	set hive.merge.smallfiles.avgsize=16000000;
	set hive.merge.mapredfiles=true;
	set hive.merge.mapfiles=true;
	
	//SET mapred.map.tasks=3;
	//set mapred.reduce.tasks=3;

REGEX_referer	(http|https)\:\/\/([a-z0-9\-_]+\.)+([a-z0-9\-_]+)\/.* http://xnuvo.com/#!/x/product_categories/-2?index=0&is_view_all=1&o rder=0&page=9&resource_type=Brand
REGEX_LOG	\\[([0-9\\-T\\+\\:]+)\\] (\\d{0,3}\\.\\d{0,3}\\.\\d{0,3}\\.\\d{0,3}) \"(GET|POST (.+) (HTTP/1.1|HTT/1.0)\" ([0-9]+) ([0-9]+) ref=\"(.+)\" ua=\"(.+)\" ck=\"(?:.*user=(\\d+)\\;.*|.*)(?:.*remember_user_token=(.+)\\;.*)\"
	"\\d{0,1}\\.\\d{0,3}\\.\\d{0,3}\\.\\d{0,3}\\s-\\s-\\s\\[(.*)\\]\\s\\\"(GET|POST)\\ s(.+)\\s(HTTP\\/1.1|HTTP\\/1.0)\"\\s([0-9]+)\\s([0-9]+)\\s\\\"(.+)\\\"\\s\\\"(. +)\\\"”
LOG	[2012-11-07T07:29:57+00:00] 1.215.74.100 "GET /x/periodicals?u=2

	HTTP/1.1" 304 0 ref="http://d1.xnuvo.com/x/weekly_features" ua="Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)" ck="nuvo_test=BAh7CEkiD3Nlc3Npb25faWQGOgZFRkkiJThhYmFiN
	Tc0YzllYjc3YWY3MzMyMzk4MzRjYmQyNWQzBjsAVEkiEF9jc3JmX3
	Rva2VuBjsARkkiMVpwN1V0UHd1ZzFyMGV6OWFjaWtGUHVYL1R1
	TUwcHJFb3R5eXRZc0xjWk09BjsARkkiGXdhcmRlbi51c2VyLnVzZXIu
	a2V5BjsAVFsISSIJVXNlcgY7AEZbBmkHSSIiJDJhJDEwJHlBVktxWC
	5SzZjOFB1TXl4bzM1OU8GOwBU--d00a417660938fa216293380c08
	44bb3b24f413; __utmc=116165651; user=2; rp=;
	__utma=116165651.338094479.1351501433.1351501433.13520708
	8.2;
	__utmz=116165651.1351501433.1.1.utmcsr=(direct)|utmccn=(direct)|
	utmcmd=(none); locale=ko-KR;
	remember_user_token=BAhbB1sGaQdJIiIkMmEkMTAkeUFWS3FYLn
	LNmM4UHVNeXhvMzU5TwY6BkVU--8e4702f6c91caf15f73ae6e60ec
	3459b3371a21; locale=ko-KR;
	fbm_258446560881949=base_domain=.xnuvo.com;
	remember_user_token=BAhbB1sGaQLyAUkiIiQyYSQxMCR4UUh4eF
	Y2OVNvWkpEL1dFRFVGSUYuBjoGRVQ%3D--626368785e0317eb3
	e18bcef9455b97562d27cf;
	fbm_343499672391458=base_domain=.xnuvo.com"

COOKIE	user remember_user_token uri = torken, user
__DEFALUT_PARTITION__ 확인	ref 값이 '-' 을 가지고 있음 from(select simpleReferer(ref) r, ref, time from dump_logs)a select * limit 1000;

2. HIVE table and logic

TBL_LOGS	create table dump(time STRING, url STRING, method STRING, uri STRING, version STRING, status STRING, size STRING, ref STRING, ua STRING, user_id STRING, user_token STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE; load data local inpath '/data/log/*' into table dump; FROM dump INSERT OVERWRITE TABLE dump SELECT logRegex(time) as time, url, method, uri, version, status, size, ref, ua, user_id, user_token; FROM dump INSERT OVERWRITE TABLE dump SELECT * where time is not null; INSERT OVERWRITE TABLE dump SELECT t2.time time, t2.url url, t2.method method, t2.uri uri, t2.version version, t2.status status, t2.size size, t2.ref ref, t2.ua ua, t1.user_id user_id, t2.user_token user_token FROM (SELECT distinct user_token, user_id FROM dump WHERE user_id is not null and user_token is not null and user_id!="" and user_token!="") t1 LEFT OUTER JOIN dump t2 ON (t1.user_token=t2.user_token); CREATE TABLE logs as SELECT d.*, u.gender user_gender FROM users u RIGHT OUTER JOIN dump d ON (u.id=d.user_id);

테스트중	BAhbB1sGaQIXDEkiliQyYSQxMCROcUNMNEcuVzJYT0U1dS9BTm 1S3plBjoGRVQ%3D--bd6ef4d66f7764bc8ba46e65569f170832fe0d72 20121116 BAhbB1sGaQLPDEkiliQyYSQxMCQyYUVaQ1U0bUx6ZFY1QnBGSk 2Wm1lBjoGRVQ%3D--188918d38e02bbc3cfaddfaf4e3cc9b6f8e7be6 6 20121113 BAhbB1sGaQLFB0kiliQyYSQxMCRkS0dDMXIveXBINUI2eDBXekt2T 9PBjoGRVQ%3D--42574040336e2d231543707769a95698469f44b7 FROM(SELECT dateToDay(time) day, simpleReferer(ref) sref, user_token, time, ref, uri, method FROM logs DISTRIBUTE BY day, user_token, sref)a select dateToSec(time) time, sref, ref, uri, method where day=20121115 and user_token='BAhbB1sGaQIpAUkiliQyYSQxMCRESi9lQ2tMVXo5d1hx SVJqWTVWVHVPBjoGRVQ%3D--9e23f30a8b6cd2ae5bf8566c07afa 1afb8df3ca' order by time; FROM(SELECT dateToDay(time) day, simpleReferer(ref) sref, user_token, time FROM logs DISTRIBUTE BY day, user_token, sref)a select day, user_token, sref, min(time) time group by day, user_token, sref order by day, time;
TBL_USERS	CREATE TABLE users(id STRING, gender STRING, facebook_user_name STRING, facebook_uid STRING, twitter_screen_name STRING, created_at STRING, updated_at STRING, email STRING, encrypted_password STRING, reset_password_token STRING,

	reset_password_sent_at STRING, remember_created_at STRING, sign_in_count STRING, current_sign_in_at STRING, last_sign_in_at STRING, current_sign_in_ip STRING, last_sign_in_ip STRING, password_salt STRING, confirmation_token STRING, confirmed_at STRING, confirmation_sent_at STRING, unconfirmed_email STRING, failed_attempts STRING, unlock_token STRING, locked_at STRING, authentication_token STRING, delivery_address_id STRING, user_role STRING, eula_version STRING, locale STRING, user_status STRING, open_brand_agreement STRING, unsubscribed_from_weekly_news_letter STRING, unsubscribed_from_notification_email STRING, unsubscribed_from_order_notification_email STRING, mobile_login_count STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE; load data local inpath '/data/mysql/dump/users.txt' into table users;

TBL_INFLOW	set hive.exec.dynamic.partition.mode=nonstrict;
	set hive.exec.dynamic.partition=true;
	SET hive.exec.max.dynamic.partitions.pernode=10000;
	SET hive.exec.max.dynamic.partitions=10000;
	SET hive.exec.max.created.files=150000;
	
	CREATE TABLE inflow
	(user_gender STRING)
	PARTITIONED BY (time STRING, ref STRING)
	ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
	STORED AS TEXTFILE;
	
	INSERT OVERWRITE TABLE inflow PARTITION(time, ref)

```
SELECT user_gender, dateToHour(time) time, simpleReferer(ref) ref
FROM logs
DISTRIBUTE BY time, ref;

INSERT OVERWRITE LOCAL DIRECTORY '/root/honeybee/inflow'
SELECT
   concat_ws(
      '\t',
      concat(
         substr(dataTbl.time, 1, 4), '-',
         substr(dataTbl.time, 5, 2), '-',
         substr(dataTbl.time, 7, 2), ' ',
         substr(dataTbl.time, 9, 2),
         ':00:00'
      ),
      dataTbl.ref,
      cast(dataTbl.totalCount as STRING),
      cast(dataTbl.femaleCount as STRING),
      cast(dataTbl.maleCount as STRING)
   )
   FROM (
      SELECT nullData.time time, nullData.ref ref, nullData.totalCount
totalCount, genderData.femaleCount femaleCount,
genderData.maleCount maleCount
         FROM (
            SELECT female.time, female.ref, female.totalCount
femaleCount, male.totalCount maleCount
            FROM (
               SELECT time, ref, count(ref) totalCount
                  FROM inflow
                  WHERE user_gender = 'female'
                  GROUP BY time, ref, user_gender
            ) female
            FULL OUTER JOIN (
               SELECT time, ref, count(ref) totalCount
               FROM inflow
               WHERE user_gender = 'male'
               GROUP BY time, ref, user_gender
            ) male
            ON female.time = male.time AND female.ref = male.ref
         ) genderData
         FULL OUTER JOIN (
```

	SELECT time, ref, count(ref) totalCount FROM inflow WHERE user_gender is null GROUP BY time, ref, user_gender) nullData ON genderData.time = nullData.time AND genderData.ref = nullData.ref) dataTbl;
TBL_LIFETIME	CREATE TABLE lifetime (day STRING, ref STRING, user_gender STRING, average_value DOUBLE) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE; INSERT OVERWRITE TABLE lifetime SELECT day, ref, user_gender, avg(min) average_value FROM (SELECT day, ref, user_token, user_gender, lifetime(min) min FROM (SELECT dateToDay(time) day, simpleReferer(ref) ref, user_token, user_gender, dateToMin(time) min FROM logs DISTRIBUTE BY day, ref, user_token, user_gender) stage1 GROUP BY day, ref, user_token, user_gender) stage2 GROUP BY day, ref, user_gender; INSERT OVERWRITE LOCAL DIRECTORY '/root/honeybee/lifetime' SELECT concat_ws('\t', concat(substr(dataTbl.day, 1, 4), '-', substr(dataTbl.day, 5, 2), '-', substr(dataTbl.day, 7, 2)), regexp_replace(dataTbl.ref, '^$', 'NULL'), cast(dataTbl.null_value as STRING), cast(dataTbl.female_value as STRING), cast(dataTbl.male_value as STRING)

	) FROM (SELECT nullData.day day, nullData.ref ref, round(nullData.average_value, 5) null_value, round(genderData.female_value, 5) female_value, round(genderData.male_value, 5) male_value FROM (SELECT female.day day, female.ref ref, female.average_value female_value, male.average_value male_value FROM (SELECT day, ref, sum(average_value) average_value FROM lifetime WHERE user_gender = 'female' GROUP BY day, ref, user_gender) female FULL OUTER JOIN (SELECT day, ref, sum(average_value) average_value FROM lifetime WHERE user_gender = 'male' GROUP BY day, ref, user_gender) male ON female.day = male.day AND female.ref = male.ref) genderData FULL OUTER JOIN (SELECT day, ref, sum(average_value) average_value FROM lifetime WHERE user_gender is null GROUP BY day, ref, user_gender) nullData ON genderData.day = nullData.day AND genderData.ref = nullData.ref) dataTbl;

잡다한것

MAP REDUCE 사용법	add jar /usr/local/src/hive-0.9.0/lib/hive-contrib-0.9.0.jar; add jar /usr/local/hive/hive_aux/honeybee-hadoop.jar; create temporary function dateToDay as 'org.honeybee.hive.DateToDay'; create temporary function simpleReferer as 'org.honeybee.hive.SimpleReferer';

	create temporary function referer as 'org.honeybee.hive.RefererReduce'; from(select dateToDay(time) day, simpleReferer(ref) ref, user_id from logs CLUSTER by day, ref limit 10) a REDUCE day, ref, user_id USING 'java -cp /usr/local/hive/hive_aux/honeybee-hadoop.jar org.honeybee.hive.lifetime.LifeTimeReduce' AS k, v; from(select * from test cluster by id) mout REDUCE id, gender USING 'java -cp honeybee-hadoop.jar org.honeybee.hive.lifetime.LifeTimeReduce' AS k, v;

3. HIVE DataFlow

1) 유입 경로 데이터(구매량, 성별)

보여주는 방식 : 가로축 : 시간 / 세로축 : 유입량, 세로축으로 표현되는 막대 그래프는 스택형식으로 유입량의 분포를 함께 표기 특성 날짜의 그래프를 선택하면 그 날의 유입량 분포를 보여주는 그래프가 오른쪽에 그려짐. 이때 분포 그래프의 양쪽에 기준일 전, 후 비교 데이터(전날 대비 증가, 감소)를 보여줌

1-1 Mapper (하루 기준)

	LOG	MySQL
input	time,user_id,time,referer...	
output	key : time(yyyy-MM-dd)	

	val : user_id, time, referer	

1-2 Reduce (하루 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd) val : user_id, time, referer	
output	key : time(yyyy-MM-dd) referer val : user_id, time	

1-3 Mapper (하루 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd) referer val : user_id, time	
output	key : time(yyyy-MM-dd) referer val : user_id, time	

1-4 Reduce (하루 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd) referer val : user_id, time	select : referer{user_id → sex}
output		insert : time(yyyy-MM-dd), referer, count, female, male

1-5 MySQL (하루 기준)

day (index)	referer	count	female	male
2012.10.01	facebook	23	23	21
2012.10.01	twitter	12	454	78
2012.10.01	etc	123	324	34

1-6 Mapper (시간 기준)

	LOG	MySQL
input	time,user_id,time,referer...	
output	key : time(yyyy-MM-dd hh) val : user_id, time, referer	

1-7 Reduce (시간 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd hh) val : user_id, time, referer	
output	key : time(yyyy-MM-dd hh) referer val : user_id, time	

1-8 Mapper (시간 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd hh) referer val : user_id, time	
output	key : time(yyyy-MM-dd hh) referer val : user_id, time	

1-9 Reduce (시간 기준)

	LOG	MySQL
input	key : time(yyyy-MM-dd hh) referer val : user_id, time	select : referer{user_id → sex}
output		insert : time(yyyy-MM-dd hh), referer, count, female, male

1-10 MySQL (시간 기준)

hour (index)	refer	count	female	male	

2012102301	facebook	23	23	21	
2012102301	twitter	12	454	78	
2012102301	etc	123	324	34	

2). Average User Lifetime(유입 경로 접속 시간 보여주기)

필요한 데이터 : 유저의 첫 로그 데이터의 Referer(여기에 유입 경로가 담겨있음), 로그 기록 시간

보여주는 방식 : 세로축(라이프타임) / 가로축(기간), 막대 그래프로 표현. 막대 그래프는 스택 형식으로 유입 경로들을 표현, 특정 날짜를 선택하면 그날의 분포도를 도넛 그래프로 보여준다. 꺽은선으로 표현하고 해당 날짜를 선택하면 텍스트로 트위터는 몇분, 페이스북은 몇분 라이프타임을 표시해준다.

2-1 Mapper

LOG	MySQL
key : sesstion_id val : refer user_id time uri	

2-2 Reduce

LOG	MySQL
key : time(하루 기준) val : sesstion_id user_id refer time lifetime uri	

2-3 Reduce

LOG	MySQL
key : refer val : time(하루 기준) sesstion_id user_id time lifetime uri	

2-4 Reduce

LOG	MySQL
key : refer val : time(하루 기준) sesstion_id user_id time lifetime (refer의 속한 평균) uri	**sex**

2-5 MySQL

day (index)	refer	lifetime	female	male	
2012.10.01	facebook	23	23	21	
2012.10.01	twitter	12	454	78	
2012.10.01	etc	12	324	34	

2-6 MySQL

hour (index)	refer	lifetime	female	male	
2012102301	facebook	23	23	21	
2012102301	twitter	12	454	78	

2012102301	etc	12	324	34	

3) 기간별 구매량

필요한 데이터 : 구매데이터,

보여주는 형식 : 가로축(기간) / 세로축(구매량), 막대 그래프로 표현.

3-1 Mapper

LOG	MySQL
	key : date (하루 기준) val : date category user_id

3-2 Reduce

LOG	MySQL
	key : category val : date (하루 기준) date user_id

3-4 Reduce

LOG	MySQL
	key : refer (HDFS, MySQL 데이터호출) val : category

	date (하루 기준) date user_id sex

3-5 MySQL 날짜별 구매

day (index)	category	refer	Purchases	female	male
2012.10.01	a	facebook	123	12	234
2012.10.02	a	twitter	12	32	3
2012.10.03	c	etc	323	12	434

3-6MySQL 시간별 구매

hour (index)	category	refer	Purchases	female	male
2012101001	a	facebook	123	12	234
2012101001	a	twitter	12	32	3
2012101001	c	etc	323	12	434

3-7MySQL 상점별 구매량

store_id	day	refer	Purchases	female	male
12323213					
34324123					
23422323					

4) 구매 액량

필요한 데이터 : 구매데이터, 구매 액수

보여주는 방식 : 가로축(구매액수), 세로축(구매건수) 막대 그래프로 표현

4-1 Mapper (날짜별 기준)

LOG	MySQL
	key : date (하루 기준) val : date price user_id store_id

4-2 Reduce (날짜별 가격 기준)

LOG	MySQL
	key : price val : date date (하루 기준) price user_id store_id

4-3 MySQL (날짜 기준)

day	price1 (0-1)	price2 (1-5)	price3 (5-10)	price4 (10-20)	price5 (20~)
2012.10.01					
2012.10.02					
2012.10.03					

4-4 Reduce (상점 기준)

LOG	MySQL
	key : store_id val : date date (하루 기준) price user_id

4-5 Reduce (상점의 일자별)

LOG	MySQL

	key : date (하루 기준) val : date store_id price user_id

4-6 MySQL (상점 기준)

store_id	day	price1 (0-1)	price2 (1-5)	price3 (5-10)	price4 (10-20)	price5 (20~)
12312	2012.10.01					
12312	2012.10.02					
12312	2012.10.03					

4-4 Reduce (유저 기준)

LOG	
	key : user_id val : date date (하루 기준) price store_id

4-5 Reduce (유저 구매 날짜 기준)

LOG	MySQL
	key : date (하루 기준) val : date store_id price user_id

4-6 MySQL

user_id	day	price1 (0-1)	price2 (1-5)	price3 (5-10)	price4 (10-20)	price5 (20~)
12312	2012.10.01					

12312	2012.10.02					
12312	2012.10.03					