故障回顾：

1. 开发误操作全表更新了配置表的信息，将 Content 字段全部更新成了一个值
2. Content 字段包含很多特殊字符，例如换行回车制表符等等。

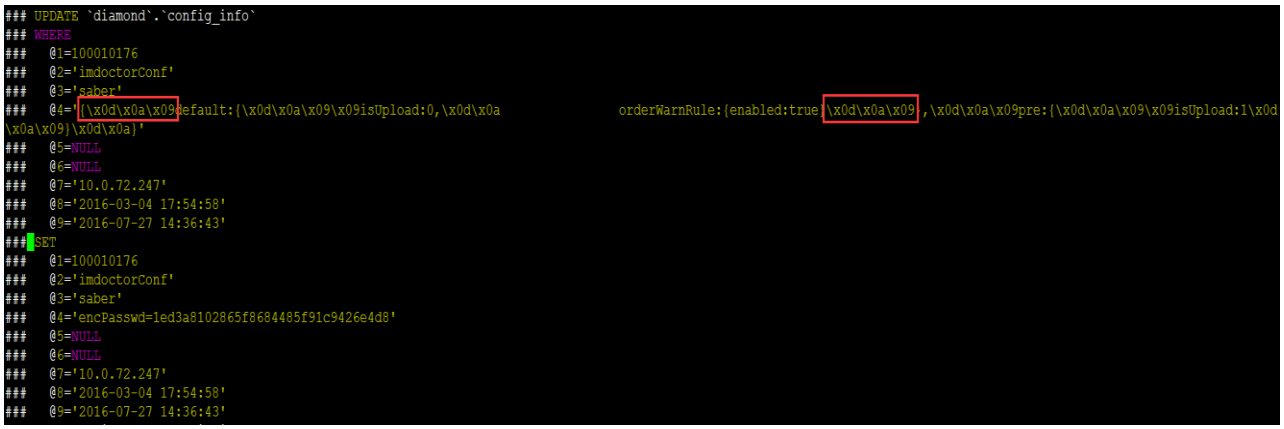
思路：

将开发操作时间段 binlog 解析成 text 文本，使用回滚脚本生成回滚语句，回滚数据库的表数据。

发现问题：

1. 使用 mysqlbinlog 解析出来的 text 文件对于特殊字符的处理存在缺陷，不能正常显示特殊符号。
2. 业务 content 包含特殊字符过于复杂，不能全面准确的替换

如下图，包含诸如此类的 windows 端存储的十六进制符



mysqlbinlog 本身对于这种字符的处理存在缺陷，导致无法生成正确的结果，从而使得回滚语句生成出现问题。

对于 MYSQL 回滚来说 \x0[x]格式的字符串 将被当做转义插入从而变成一个非法字符（x0[x]）存在于表中

而 mysqldump 对于这种问题的处理是没有问题的如下图

```
LOCK TABLES `config_info` WRITE;
/*!40000 ALTER TABLE `config_info` DISABLE KEYS */;
INSERT INTO `config_info` VALUES (100010176,'imdoctorConf','saber','{\r\n   default:{\r\n       isUpload:0,\r\n              orderWarnRule:{enabled:true}\r\n },\r\n  pre:
{\r\n       isUpload:1\r\n  }\r\n}',NULL,NULL,'10.0.72.247','2016-03-04 17:54:58','2016-07-27 14:36:43');
/*!40000 ALTER TABLE `config_info` ENABLE KEYS */;
UNLOCK TABLES;
```

可以看到 mysqldump 出来的 SQL 文件对于这种字符的处理完全是按照 linux 端的形式展示的

下面放一张 ascii 码对照表

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Linux ascii 对照表

```
1.  C program '\X' escapes are noted.
2.
3.         Oct   Dec   Hex   Char                           Oct   Dec   Hex   Char
4.         ------------------------------------------------------------------------
    --
5.         000   0     00    NUL '\0'                       100   64    40    @
6.         001   1     01    SOH (start of heading)         101   65    41    A
7.         002   2     02    STX (start of text)            102   66    42    B
8.         003   3     03    ETX (end of text)              103   67    43    C
9.         004   4     04    EOT (end of transmission)      104   68    44    D
10.        005   5     05    ENQ (enquiry)                  105   69    45    E
11.        006   6     06    ACK (acknowledge)              106   70    46    F
```

12.	007	7	07	BEL '\a' (bell)	107	71	47	G	
13.	010	8	08	BS '\b' (backspace)	110	72	48	H	
14.	011	9	09	HT '\t' (horizontal tab)	111	73	49	I	
15.	012	10	0A	LF '\n' (new line)	112	74	4A	J	
16.	013	11	0B	VT '\v' (vertical tab)	113	75	4B	K	
17.	014	12	0C	FF '\f' (form feed)	114	76	4C	L	
18.	015	13	0D	CR '\r' (carriage ret)	115	77	4D	M	
19.	016	14	0E	SO (shift out)	116	78	4E	N	
20.	017	15	0F	SI (shift in)	117	79	4F	O	
21.	020	16	10	DLE (data link escape)	120	80	50	P	
22.	021	17	11	DC1 (device control 1)	121	81	51	Q	
23.	022	18	12	DC2 (device control 2)	122	82	52	R	
24.	023	19	13	DC3 (device control 3)	123	83	53	S	
25.	024	20	14	DC4 (device control 4)	124	84	54	T	
26.	025	21	15	NAK (negative ack.)	125	85	55	U	
27.	026	22	16	SYN (synchronous idle)	126	86	56	V	
28.	027	23	17	ETB (end of trans. blk)	127	87	57	W	
29.	030	24	18	CAN (cancel)	130	88	58	X	
30.	031	25	19	EM (end of medium)	131	89	59	Y	
31.	032	26	1A	SUB (substitute)	132	90	5A	Z	
32.	033	27	1B	ESC (escape)	133	91	5B	[	
33.	034	28	1C	FS (file separator)	134	92	5C	\	'\\ '
34.	035	29	1D	GS (group separator)	135	93	5D	]	
35.	036	30	1E	RS (record separator)	136	94	5E	^	
36.	037	31	1F	US (unit separator)	137	95	5F	_	
37.	040	32	20	SPACE	140	96	60	`	
38.	041	33	21	!	141	97	61	a	
39.	042	34	22	"	142	98	62	b	
40.	043	35	23	#	143	99	63	c	
41.	044	36	24	$	144	100	64	d	
42.	045	37	25	%	145	101	65	e	
43.	046	38	26	&	146	102	66	f	
44.	047	39	27	?	147	103	67	g	
45.	050	40	28	(	150	104	68	h	
46.	051	41	29	)	151	105	69	i	
47.	052	42	2A	*	152	106	6A	j	
48.	053	43	2B	+	153	107	6B	k	
49.	054	44	2C	,	154	108	6C	l	
50.	055	45	2D	-	155	109	6D	m	
51.	056	46	2E	.	156	110	6E	n	
52.	057	47	2F	/	157	111	6F	o	
53.	060	48	30	0	160	112	70	p	
54.	061	49	31	1	161	113	71	q	

055.	062	50	32	2	162	114	72	r
056.	063	51	33	3	163	115	73	s
057.	064	52	34	4	164	116	74	t
058.	065	53	35	5	165	117	75	u
059.	066	54	36	6	166	118	76	v
060.	067	55	37	7	167	119	77	w
061.	070	56	38	8	170	120	78	x
062.	071	57	39	9	171	121	79	y
063.	072	58	3A	:	172	122	7A	z
064.	073	59	3B	;	173	123	7B	{
065.	074	60	3C	<	174	124	7C	|
066.	075	61	3D	=	175	125	7D	}
067.	076	62	3E	>	176	126	7E	~
068.	077	63	3F	?	177	127	7F	DEL

四种方法解决这个问题：

1. 采用 BINLOG_DUMP 协议源生获取 binlog 内容，跳过对特殊字符解析的过程。

优点：源生解析等于 slave binlog 回放，准确率高

缺点：速度较慢 可能恢复时间较长

2. Restore 备份，通过 binlog 恢复到指定时间

优点：能恢复到任意指定时间点，相对安全 不会对线上系统产生影响

缺点：速度慢，需要 binlog 保存完整

3. 采用 mysqlbinlog 工具解析导入再通过 replace 函数全局替换

优点：方便，采用源生 mysqlbinlog 解析

缺点：错误率高，可能遗漏部分特殊字符，需要业务端检查

4. 采用自研工具在解析阶段直接解析出相关字符串

优点：灵活，可以自定义解析维度

缺点：成本较高，需要完整的开发测试资源

推荐两个 GITHUB 上的项目：

https://github.com/danfengcao/binlog2sql -- 基于 binlog dump 协议开发

https://github.com/58daojia-dba/mysqlbinlog_flashback

试用 可以解决上述需求

```
#python binlog2sql.py  -uroot -p'roda3983434aLRDDdfjot' -P 3380 -h 127.0.0.1   --start-file='mysql-bin.000001'
USE diamond;
create table liuyang1(name varchar(10));
INSERT INTO `diamond`.`liuyang1`(`name`) VALUES ('我的\n'); #start 437 end 612 time 2017-08-12 22:03:20
INSERT INTO `diamond`.`liuyang1`(`name`) VALUES ('我的    \r\n'); #start 643 end 822 time 2017-08-12 22:03:43
DELETE FROM `diamond`.`liuyang1` WHERE `name` IS NULL LIMIT 1; #start 853 end 1042 time 2017-08-12 22:03:55
DELETE FROM `diamond`.`liuyang1` WHERE `name`='我的\n' LIMIT 1; #start 853 end 1042 time 2017-08-12 22:03:55
DELETE FROM `diamond`.`liuyang1` WHERE `name`='我的    \r\n' LIMIT 1; #start 853 end 1042 time 2017-08-12 22:03:55
INSERT INTO `diamond`.`liuyang1`(`name`) VALUES ('我的    \r\n'); #start 1073 end 1252 time 2017-08-12 22:03:59
UPDATE `diamond`.`liuyang1` SET `name`='wo' WHERE `name`='我的    \r\n' LIMIT 1; #start 1283 end 1467 time 2017-08-12 22:04:54
UPDATE `diamond`.`liuyang1` SET `name`='wo      ' WHERE `name`='wo' LIMIT 1; #start 1498 end 1675 time 2017-08-12 22:05:51
```