DB2 Temporal摘要

> https://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0058481.html
>
> https://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/

时间概念

1. 有效时间，BUSINESS_TIME，区间，用户给出，左闭右开
2. 事务时间，SYSTEM_TIME，区间，系统给出，未明确定义开闭
    1. ROW BEGIN，记录被修改的语句开始执行的时间：
        1. 同一事务内，先后INSERT、UPDATE同一行，row_begin_col记录为INSERT开始的时间

```
BEGIN;
INSERT INTO t VALUES (...); //2011-01-01 00:00:00
UPDATE t SET ...; //2011-01-01 00:00:02
COMMIT
-------------------------------------------------
这条记录的row_begin_col为2011-01-01 00:00:00
```

        2. 同一事务内，写不同行，所有行的row_begin_col都一样

```
BEGIN;
INSERT INTO t VALUES(1, ...) //2011-01-01 00:00:00
INSERT INTO t VALUES(2, ...) //2011-01-01 00:00:02
INSERT INTO t VALUES(3, ...) //2011-01-01 00:00:04
COMMIT
-------------------------------------------------
三条记录的row_begin_col为2011-01-01 00:00:00
```

    2. ROW END，语句不再是“当前行”（current）的时候
3. 事务开始时间，TRANSACTION START ID，叫ID，其实还是TIMESTAMP类型；

   由系统给出。如果一个事务中有多条INSERT、UPDATE，这些记录的transaction_start_id都为这个事务的开始时间，用于查看多条记录是不是在同一个事务中被修改的
4. `ROW BEGIN`和`TRANSACTION START ID`有什么不同？

   一般情况下，这两个值是一样的；
   https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/rzahf/rzahftmprlconflicts.htm#rzahftemporaltable描述了什么时候这两个值不一样：

Current Timestamp	Transaction A	Trans A SYS_START value	Transaction B	Trans B SYS_START value
T1	`INSERT INTO policy_info (policy_id, coverage) VALUES ('S777',7000)`	T1		
T2			`INSERT INTO policy_info (policy_id, coverage) VALUES ('T888',8000)`	T2
T3			`COMMIT`	T2
T4	`UPDATE policy_info SET policy_id = 'X999' WHERE policy_id = 'T888'`	T1		
T5	`COMMIT`	T1		

我们关注T888这条记录，TransB在T2时刻先提交，所以(row_begin_col, row_end_col, trans_start_id) = (T2, UC, T2)；TransA在T1时刻提交，更新了T888这条记录，旧记录入历史表，添加新记录到原表，旧记录(row_begin_col, row_end_col, trans_start_id)=(T2, T1, T2) **这里矛盾了，row_end_col < row_begin_col**，新记录(row_begin_col, row_end_col, trans_start_id)=(T1, UC, T1)；

row_end_col < row_begin_col矛盾出现时，DB2提供两种策略：

1. ERROR：报错
2. ADJUST，三步走
    1. 冲突发生在历史表，所以在历史表中修改row_end_col= row_begin_col+ 1ms
    2. 随历史表row_end_col的改变，原表当前记录的row_begin_col也要修改，为历史表对应记录的row_end_col值

        原表修改前（row_begin_col, row_end_col, trans_start_id）=（T1, UC, T1）

        原表修改后（row_begin_col, row_end_col, trans_start_id）=（T1+1ms, UC, T1）**不同发生了**
    3. 报警告

CREATE TABLE

整体思路类似于SQL Server 2016，两张表，原表和历史表绑定

1. 创建原表

```
CREATE TABLE policy_info
(
 policy_id    CHAR(4) NOT NULL,
 coverage     INT NOT NULL,
 bus_start    DATE NOT NULL,
 bus_end      DATE NOT NULL,
 sys_start    TIMESTAMP(12) NOT NULL
              GENERATED ALWAYS AS ROW BEGIN,
 sys_end      TIMESTAMP(12) NOT NULL
              GENERATED ALWAYS AS ROW END,
 ts_id        TIMESTAMP(12) NOT NULL
              GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
```

```
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

2. 创建历史表

   创建一个和原表有相同列、列定义的表

```
CREATE TABLE hist_policy_info
(
 policy_id    CHAR(4) NOT NULL,
 coverage     INT NOT NULL,
 bus_start    DATE NOT NULL,
 bus_end      DATE NOT NULL,
 sys_start    TIMESTAMP(12) NOT NULL,
 sys_end      TIMESTAMP(12) NOT NULL,
 ts_id        TIMESTAMP(12)
) in hist_space;
或者
CREATE TABLE hist_policy_info LIKE policy_info in hist_space;
```

3. 绑定原表和历史表

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

4. **可选**创建主键，包含BUSINESS_TIME

```
CREATE UNIQUE INDEX ix_policy
   ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

5. 隐藏SYSTEM_TIME

```
CREATE TABLE policy_info
(
 policy_id    CHAR(4) NOT NULL,
 coverage     INT NOT NULL,
 bus_start    DATE NOT NULL,
 bus_end      DATE NOT NULL,
 sys_start    TIMESTAMP(12) NOT NULL
              GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
 sys_end      TIMESTAMP(12) NOT NULL
              GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
 ts_id        TIMESTAMP(12)
              GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
 PERIOD SYSTEM_TIME   (sys_start, sys_end)
) in policy_space;
```

   通过CREATE LIKE创建历史表，历史表同样隐藏SYSTEM_TIME列

INSERT

1. INSERT会自动检测valid_start_time < valid_end_time（不是<=）；如果定义了`WITHOUT OVERLAP`，还会检测有效时间会不会重叠
2. INSERT不会记录到历史表
3. 举例

```
INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)  //bus_start, bus_end为有效时间
   VALUES('A123',12000,'2008-01-01','2008-07-01');
```

执行上述INSERT后：原表

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	transaction_start_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

历史表：空

UPDATE

1. 支持传统的UPDATE；扩展语法，支持UPDATE一段有效时间内的记录
2. UPDATE会记录原记录到历史表
3. 举例

UPDATE之前，历史表空，原表

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

1. 更新有效时间，普通的UPDATE，不支持`FOR PORTION OF BUSINESS_TIME`

```
UPDATE policy_info
  SET bus_start='2008-03-01'
  WHERE policy_id = 'B345'
  AND coverage = 18000;
```

更新后原表该行变为（省略其他行）：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

历史表中增加一条记录：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

2. 在有效时间段内更新字段，`FOR PORTION OF BUSINESS_TIME`

    1. 指定时间段恰为某行的bus_start ~ bus_end

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '2009-01-01'
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

更新后原表该行变为(省略其他行)：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

历史表中增加一条记录：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

    2. 指定时间段横跨两行的bus_start ~ bus_end

> r1.bus_start= 2001-1-1, r1.bus_end= 2001-3-1;
>
> r2.bus_start= 2001-3-1, r2.bus_end= 2001-5-1;
>
> SELECT ... FOR PORTION OF BUSINESS_TIME FROM '2001-2-1' TO '2001-4-1'

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-01' TO '2008-08-01'
   SET coverage = 14000
   WHERE policy_id = 'A123';
```

更新后原表的行(省略无关行)：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

历史表中增加记录：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

DELETE

1. 支持传统的DELETE；扩展语法，支持DELETE一段有效时间内的记录
2. DELETE会记录原记录到历史表
3. 举例

   DELETE前，原表：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

   1. 传统DELETE略
   2. 在有效时间段内删除记录，`FOR PORTION OF BUSINESS_TIME`

```
DELETE FROM policy_info
    FOR PORTION OF BUSINESS_TIME FROM '2008-6-15' TO '2008-8-15'
    WHERE policy_id= 'A123';
```

   删除后，原表：（**时态运算的体现**）

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000

   历史表增加记录：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

QUERY

1. 指定valid time period，可以查询当前、历史、将来(将来有效)的值
2. 可以在`FROM`子句指定`FOR BUSINESS_TIME, FOR SYSTEM_TIME`
3. 三种时态运算：
    1. AS OF value1，查询时间点，begin_time <= value1 && end_time > value1
    2. FROM value1 TO value2，查询时间段，begin_time >= value1 && end_time < value2，begin_time~end_time严格位于value1~value2之内(类比子集的概念)
    3. BETWEEN value1 AND value2，查询时间段，begin_time >= value1 && end_time < value2，begin_time~end_time有时间点落在value1!value2之内(类比交集的概念)
    4. 上述三种时态运算，同时适用于有效时间和事务时间
4. 举例：

    原表：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000

    历史表：

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

    1. 传统查询

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   where policy_id = 'A123'
--——————————result——————————--
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15

A123, 16000, 2008-08-15, 2009-01-01
```

不指定`FOR SYSTEM_TIME`只查询原表

2. 查询`FOR SYSTEM_TIME FROM...TO`

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
--————————————result————————————--
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
A123, 12000, 2008-01-01, 2008-07-01
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-06-01, 2008-07-01
A123, 14000, 2008-07-01, 2008-08-01
A123, 16000, 2008-08-01, 2009-01-01
```

指定`FOR SYSTEM_TIME`，查询原表、历史表

3. 查询`FOR BUSINESS_TIME AS OF`

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   where policy_id = 'A123'
--————————————result————————————--
NULL
```

没有FOR SYETEM_TIME不查询历史表，原表没有在“2008-07-15”有效的记录

4. 查询`FOR BUSINESS_TIME AS OF, FOR SYSTEM_TIME FROM...TO`

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
--————————————result————————————--
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-07-01, 2008-08-01
```

5. 官方举例到此为止