SQL：2011 时态数据库相关

1. 时间不采用创建新的period类型的方式，而是和普通列一样的存在；定义一个*period definitions*，作为元数据，记录start和end列
2. 时间区间为左闭右开，[start, end)
3. 有效时间区域*application-time period*，名字由用户任意指定，每张表最多有一个有限时间区间

   举例：

   1. 建表

      ```
      CREATE TABLE Emp(
          ENo INTEGER,
          EStart DATE,
          EEnd DATE,
          EDept INTEGER,
          PERIOD FOR EPeriod (EStart, EEnd)
      )
      ```

      有效时间列名字可以任意指定，但类型必须是`DATE`或`timestamp`，有效时间这两列的类型必须一致

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-02-03	3
22217	2011-02-03	2011-09-10	4
22217	2011-09-10	2011-11-12	3

   2. INSERT

      ```
      INSERT INTO Emp
          VALUES (22217,
          DATE ‘2010-01-01’,
          DATE '2011-11-12', 3)
      ```

      传统SQL标准即支持

   3. UPDATE、DELETE

      1. 传统SQL标准的UPDATE、DELETE依然有效，并且可以UPDATE、DELETE `application-time period`
      2. 扩展SQL语法指定`application-time period` [start, end) 范围内数据的UPDATE和DELETE

```
UPDATE Emp
    FOR PORTION OF EPeriod
        FROM DATE '2011-02-03'
        TO DATE '2011-09-10'
SET EDept = 4
WHERE ENo = 22217
```

```
DELETE Emp
    FOR PORTION OF EPeriod
        FROM DATE '2011-02-03'
        TO DATE '2011-09-10'
WHERE ENo = 22217
```

关键字 **PORTION OF EPeriod**

4. 主键约束
    1. 历史数据库表的主键，除用户定义主键外，必须包含`application-time period` EStart和EEnd
    2. 大多数情况下，主键还不充分，如

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-09-10	3
22217	2010-02-03	2011-11-12	4

    虽然满足主键的唯一性，但是`application-time period`存在交集，不满足有效时间的唯一性约束
    3. 可以扩展语法，用于是否进行有效时间的唯一性约束

```
ALTER TABLE Emp
ADD PRIMARY KEY (ENo,
    EPeriod WITHOUT OVERLAPS)
```

5. 外键约束
    1. 假设有另外一张Dept表：

```
CREATE TABLE Dept(
    DNo INTEGER,
    DStart DATE,
    DEnd DATE,
    DName VARCHAR(30),
    PERIOD FOR DPeriod (DStart, DEnd),
    PRIMARY KEY (DNo,
        DPeriod WITHOUT OVERLAPS)
)
```

假设Emp数据为：

ENo	EStart	EEnd	EDept
22218	2010-01-01	2011-02-03	3
22218	2010-02-03	2011-11-12	4

Dept数据为：

DNo	DStart	DEnd	DName
3	2009-01-01	2011-12-31	Test
4	2011-06-01	2011-12-31	QA

我们在Emp.EDept和Dept.DNo上建立外键约束，对于普通的数据库模型，是符合外键约束的；但对于时态数据库不符合，在有效时间上存在逻辑矛盾：Emp表显示22218号员工在2010-02-03-2011-11-12住在4号公寓，但DEpt表显示4号公寓有效时间为2011-06-01-2011-12-31。

2. 扩展语法，提供外键的有效时间一致性检测

```
ALTER TABLE Emp
ADD FOREIGN KEY
    (Edept, PERIOD EPeriod)
    REFERENCES Dept
    (DNo, PERIOD DPeriod)
```

6. 查询`application-time period`表
    1. 当然，可以用WHERE条件，如

    ```
    查询22217号员工在2011年1月2日工作的department：
    SELECT Name, Edept
    FROM Emp
        WHERE ENo = 22217
        AND EStart <= DATE '2011-01-02'
        AND EEnd > DATE '2011-01-02'
    ```

    ```
    查询22217号员工在2010-1-1至2011-1-1期间就职的department：
    SELECT Ename, Edept
    FROM Emp
        WHERE ENo = 22217
        AND EStart < DATE '2011-01-01'
        AND EEnd > DATE '2010-01-01'
    ```

    2. SQL：2011规定了包含period的描述，`CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES, IMMEDIATELY SUCCEEDS`，对应于6.1的例子的描述：

```
SELECT Ename, Edept
FROM Emp
    WHERE ENo = 22217 AND
    EPeriod CONTAINS DATE '2011-01-02'
```

```
SELECT Ename, Edept
FROM Emp
WHERE ENo = 22217 AND
    EPeriod OVERLAPS
    PERIOD (DATE '2010-01-01',
    DATE '2011-01-01')
```

4. 事务时间区域叫做*system-time period*，`SYSTEM_TIME`，每张表最多有一个事务时间区间，区间的首、尾可由用户定义列名，如

```
CREATE TABLE Emp
    ENo INTEGER,
    Sys_start TIMESTAMP(12) GENERATED
        ALWAYS AS ROW START,
    Sys_end TIMESTAMP(12) GENERATED
        ALWAYS AS ROW END,
    EName VARCHAR(30),
    PERIOD FOR SYSTEM_TIME (Sys_start,
        Sys_end)
) WITH SYSTEM VERSIONING
```

Sys_start 和 Sys_end（可由用户定义这两个名字）类型要保持一致DATE或者timestamp，大多数实现为timestamp类型；同样为左闭右开的时间区间 [Sys_start, Sys_end)

1. Sys_start和Sys_end用户不可修改，只能被DBMS赋值，这也是定义这两列时声明`GENERATED ALWAYS`的原因
2. 对回滚表的INSERT将自动地把Sys_start列赋值为*transaction timestamp*，Sys_end赋值为该列类型最大值；如

```
INSERT INTO Emp (ENo, EName) VALUES (22217, 'Joe')
```

ENo	Sys_start	Sys_end	EName
22217	2012-01-01 09:00:00	9999-12-31 23:59:59	Joe

3. 对回滚表，UPDATE，DELETE只能操作当前行，不能操作历史行；UPDATE，DELETE会导致自动产生一条INSERTION到回滚表中

```
UPDATE Emp
    SET EName = 'Tom'
    WHERE ENo = 22217
```

ENo	Sys_start	Sys_end	EName
22217	2012-01-01 09:00:00	2012-02-03 10:00:00	Joe
22217	2012-02-03 10:00:00	9999-12-31 23:59:59	Tom

```
DELETE FROM Emp
    WHERE ENo = 22217
```

ENo	Sys_start	Sys_end	EName
22217	2012-01-01 09:00:00	2012-02-03 10:00:00	Joe
22217	2012-02-03 10:00:00	2012-06-01 00:00:00	Tom

4. 回滚表的主键约束、外键约束

   回滚表的主键无需引入事务时间列

   如：

```
ALTER TABLE Emp
    ADD PRIMARY KEY (ENo);

ALTER TABLE Emp
    ADD FOREIGN KEY (Edept)
    REFERENCES Dept (DNo)
```

   SQL标准给出的解释是

   > Any constraints that were in effect when a historical system row was created would have already been checked when that row was a current system row, so there is never any need to enforce constraints on historical system rows.

5. 回滚表的查询

   SQL标准规定了三种语法扩展

   1. FOR SYSTEM_TIME AS OF

      查询事务时间点

```
SELECT ENo,EName,Sys_Start,Sys_End
    FROM Emp FOR SYSTEM_TIME AS OF
    TIMESTAMP '2011-01-02 00:00:00'
```

      上述查询的含义：Sys_start_time<= 2011-01-02 00:00:00 && Sys_end_time>2011-01-02 00:00:00

   2. FOR SYSTEM_TIME FROM ... TO

查询一段事务时间，左闭右开

```
SELECT ENo,EName,Sys_Start,Sys_End
    FROM Emp FOR SYSTEM_TIME FROM
    TIMESTAMP '2011-01-02 00:00:00’TO
    TIMESTAMP '2011-12-31 00:00:00'
```

上述查询的含义：Sys_start_time>=2011-01-02 00:00:00 && Sys_end_time<2011-12-31 00:00:00

3. FOR SYSTEM_TIME BETWEEN ... AND

   查询一段事务时间，闭区间

```
SELECT ENo,EName,Sys_Start,Sys_End
    FROM Emp FOR SYSTEM_TIME BETWEEN
    TIMESTAMP '2011-01-02 00:00:00'AND
    TIMESTAMP '2011-12-31 00:00:00'
```

   上述查询的含义：Sys_start_time>=2011-01-02 00:00:00 && Sys_end_time<=2011-12-31 00:00:00

4. 如果对一个回滚表的查询，没有指定上述三种语法，那么查询当前记录，而非历史记录

```
SELECT ENo,EName,Sys_Start,Sys_End FROM Emp
```

5. 查询所有记录，包括历史和当前

```
SELECT ENo,EName,Sys_Start,Sys_End
    FROM Emp FOR SYSTEM_TIME FROM
    TIMESTAMP '0001-01-01 00:00:00' TO
    TIMESTAMP '9999-12-31 23:59:59'
```

5. 双时态表

   1. 创建双时态表

```
CREATE TABLE Emp(
    ENo INTEGER,
    EStart DATE,
    EEnd DATE,
    EDept INTEGER,
    PERIOD FOR EPeriod (EStart, EEnd),
    Sys_start TIMESTAMP(12) GENERATED
    ALWAYS AS ROW START,
    Sys_end TIMESTAMP(12) GENERATED
    ALWAYS AS ROW END,
    EName VARCHAR(30),
    PERIOD FOR SYSTEM_TIME
    (Sys_start, Sys_end),
    PRIMARY KEY (ENo,

        EPeriod WITHOUT OVERLAPS),
```

```
    FOREIGN KEY
        (Edept, PERIOD EPeriod)
    REFERENCES Dept
        (DNo, PERIOD DPeriod)
) WITH SYSTEM VERSIONING
```

2. INSERT时，`system-time period` Sys_start_time为当前事务时间，Sys_end_time为时间类型支持的最大值；

   UPDATE、DELETE，支持传统的UPDATE、DELETE，和valid time的更新和删除；

   UPDATE、DELETE只适用于当前时间的行，并且UPDATE、DELETE记录一条历史记录

3. 双时态表的查询

   结合回滚表和历史表的语法

```
SELECT ENo, EDept
    FROM Emp FOR SYSTEM_TIME AS OF
    TIMESTAMP '2011-07-01 00:00:00'
    WHERE ENo = 22217 AND
    EPeriod CONTAINS DATE '2010-12-01'
```

6. 未来方向
   1. Support for period joins
   2. Support for period aggregates and period grouped queries
   3. Support for period UNION, INTERSECT and EXCEPT operators
   4. Support for period normalization
   5. Support for multiple application-time periods per table
   6. Support for non-temporal periods
7. 和以前的提案的比较
   1. 以前：时态表中的时间区间列都是无名、隐藏的，比如SELECT * 是看不到的，如果用户想获取这些列，不得不去调用特别内建的函数
   2. SQL：2011提供清晰定义的语法最小集
   3. 以前：不指定period（valid time、transaction time），默认为修改当前行，transaction time和valid time都按照当前时间来；对于回滚表，是没问题的；对历史表，valid_start_time只能是当前时间(因为period列隐藏，不能指定值)，失去了valid time指代过去、未来的意义
   4. 以前：必须通过`AS TRNSACTION TIME/ASVALIDTIME`来表明是回滚表/历史表，想增加新的period类型(比如unvalid time...)，就得再扩展语法；SQL：2011无需新的语法扩展