УДК 004.652.4 DOi 10.30837/ bi.2021.1(96).13

Т.К. Михневич[1], **О.О. Мазурова**[2]

[1]магістрант кафедри програмної інженерії ХНУРЕ,
м. Харків, tetiana.mykhnevych@nure.ua, ORCID ID 0000-0001-6377-2145

[2] кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ,
м. Харків, oksana.mazurova@nure.ua, ORCID ID 0000-0003-3715-3476

ДОСЛІДЖЕННЯ МЕТОДІВ ПІДТРИМКИ ТЕМПОРАЛЬНОСТІ В РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ

Проведено аналіз області побудови темпоральних інформаційних систем та виявлені існуючі проблеми, які потребують рішення. Запропоновано математичну модель представлення темпоральних даних з урахуванням дійсного та транзакційного часу. Розглянуто способи підтримки темпоральності у реляційних СУБД, виявлені їх переваги та недоліки. Наведено результати експериментального порівняння реалізацій підтримки темпоральності на рівні бази даних та на рівні додатку з використанням реляційних систем управління базами даних MS SQL Server та Oracle. Порівняння проводилося за метриками часу виконання різнотипних запитів та об'єму диску, що використовувався. На основі результатів дослідження методів підтримки темпоральності розроблено рекомендації стосовно організації та роботи з темпоральними моделями даних при використанні реляційних СУБД.

БІТЕМПОРАЛЬНА МОДЕЛЬ ДАНИХ, ДІЙСНИЙ ЧАС, РЕЛЯЦІЙНА СУБД, ТЕМПОРАЛЬНА МОДЕЛЬ ДАНИХ, ТЕМПОРАЛЬНІСТЬ, ТРАНЗАКЦІЙНИЙ ЧАС, ЧАСОВА МІТКА.

Михневич Т.К. , Мазурова О.А. Исследование методов поддержки темпоральности в реляционных базах данных. Проведен анализ области построения темпоральных информационных систем и выявлены существующие проблемы, требующие решения. Предложена математическая модель представления темпоральных данных с учетом действительного и транзакционного времени. Рассмотрены способы поддержки темпоральности в реляционных СУБД, выявлены их преимущества и недостатки. Приведены результаты экспериментального сравнения реализаций поддержки темпоральности на уровне базы данных и на уровне приложения с использованием реляционных систем управления базами даннях MS SQL Server и Oracle. Сравнения проводились на основании метрик времени выполнения разнообразних запросив и занятого объема диска. На основе результатов исследования методов поддержки темпоральности разработаны рекомендации по организации и работе с темпоральными моделями данных при использовании реляционных СУБД.

БИТЕМПОРАЛЬНАЯ МОДЕЛЬ ДАННЫХ, ДЕЙСТВИТЕЛЬНОЕ ВРЕМЯ, РЕЛЯЦИОННАЯ СУБД, ТЕМПОРАЛЬНАЯ МОДЕЛЬ ДАННЫХ, ТЕМПОРАЛЬНОСТЬ, ТРАНЗАКЦИОННОЕ ВРЕМЯ, ВРЕМЕННАЯ МЕТКА.

Mykhnevych T., Mazurova O. Research of Methods of Temporality Support in Relational Databases. The analysis of temporal information systems construction area is performed and the existing problems that need to be solved are identified. There was proposed a mathematical model for the presentation of temporal data, considering the valid and transactional time. Methods of supporting temporality in relational DBMS are investigated, their advantages and disadvantages are revealed. The stages of temporal database design are proposed. The results of an experimental comparison of the implementations of temporality support at the database level and at the application level using the relational database management systems MS SQL Server and Oracle are presented. Comparisons were made based on metrics of the execution time of various requests and the occupied disk space. Recommendations are developed for organizing and working with temporal data models when using relational DBMS based on an study of temporality support.

BITEMPORAL DATA MODEL, CURRENT TIME, RELATIONAL DBMS, TEMPORAL DATA MODEL, TEMPORALITY, TRANSACTION TIME, TIMESTAMP.

Вступ

На сьогодні реляційні бази даних (БД) поступово втрачають свої позиції основної концепції зберігання даних. Традиційно реляційна СУБД замінює старі значення атрибутів новими та не бере до уваги попередні стани об'єктів. Такий підхід може бути доречним не в усіх системах. Існує велика кількість предметних галузей, де зміна кожного об'єкта є критичною. У таких випадках необхідно обробляти дані, що змінюються, накопичувати історію їх зміни та видавати стан системи на будь-який момент часу. Однією із сфер, де застосовуються темпоральні моделі даних, є системи, що забезпечують підтримку вибору споживача, надаючи персоналізований перелік товарів та послуг. Вони дозволяють побудувати часову модель поведінки користувачів, що описує еволюцію вимог користувача до товарів, пропонованих системою [1]

Темпоральні (або часові) дані – це будь-які дані, що пов'язані з певними датами або проміжками часу. Для управління та зберігання таких даних краще були б пристосовані темпоральні СУБД. Але незважаючи на те, що підтримка темпоральності має свої переваги та актуальність, досі не існує єдиного стандарту реалізації темпоральних БД та майже відсутні дійсно темпоральні СУБД.

Майже всі додатки, що використовують реляційні БД (РБД), є темпоральними за своєю суттю. У них зберігаються дані, що змінюються з плином часу, але в мові запитів до РБД SQL не має адекватної і ефективної підтримки роботи саме з темпоральними даними.

Тому майже у всіх БД підтримка роботи з темпоральними даними забезпечується зусиллями програмістів. Відповідно темпоральні принципи в цих системах реалізуються по різному та не завжди ефективно, що потребує додаткового дослідження таких методів підтримки темпоральності.

1. Аналіз існуючих методів підтримки темпоральності

Зазвичай побудова БД з елементами темпоральності засновується на реляційних СУБД (РСУБД), як певна надбудова над нею [2]. Цей спосіб є найбільш простим та доступним як для розробників, так і для користувачів, адже на користь реляційних СУБД свідчать:

– успішність їх використання впродовж кількох десятиліть та надійність;

– гарна підтримка виробника, постійне вдосконалення та розвиток;

– найкраща підтримка цілісності даних;

– наявність загальноприйнятих стандартів та інше.

Але у способі використання РСУБД з підтримкою темпоральності можна виділити і наступні проблеми:

– відсутність єдиного стандарту для реалізації підтримки темпоральності; мова запитів SQL не пристосовані для роботи з темпоральними даними [2];

– РСУБД не підтримує семантику часового поля відношень, не контролює коректність його значень.

Окрім того, існує ціла низка характеристик, яким повинні задовольняти засоби підтримки темпоральності:

– специфікація періодів;

– таблиці з дійсним/транзакційним часом;

– темпоральна поведінка UPDATE / DELETE;

– темпоральні обмеження цілісності;

– предикати та функції для періодів.

Отже, можна зробити висновок, що реалізація темпоральних СУБД на базі реляційних є найбільш ефективною на сьогоднішній момент, але потребує глибокої переробки моделі даних з урахуванням темпоральної технології та особливостей обраних для реалізації РСУБД.

З іншого боку, в стандарті SQL:2011 [4] з'явилась важлива нова функціональність створювати та керувати темпоральними таблицями. Але у стандарті є і певні обмеження, а саме:

– періоди – це лише відкриті-закриті інтервали [X, Y); але це не новий тип даних, по факту це два стовпця для позначення початку та кінця періоду;

– у таблицях може бути щонайбільше один транзакційний і один дійсний період часу;

– немає понять "NOW", "UC", "infinity", "empty";

– багато відмінностей між таблицями періодів транзакційного та дійсного часу;

– обмежена підтримка формулювання темпоральних запитів.

Та навіть цей стандарт реляційні СУБД не підтримують повністю, а лише частково. При цьому темпоральні принципи в цих системах використовуються неефективно:

– неекономно використовується дисковий простір;

– низька ефективність взаємодії з темпоральними об'єктами;

– погана розширюваність БД;

– висока трудомісткість розробки БД;

– складність підтримки цілісності даних;

– складність організації доступу до даних;

– проблеми в забезпеченні конфіденційності даних.

Отже, можна виділити декілька принципових підходів до підтримки темпоральності під час використання РСУБД:

– реалізація темпоральной підтримки на рівні БД за допомогою вбудованої підтримки (вибір СУБД з максимальною кількістю темпоральних функцій);

– реалізація підтримки на рівні БД шляхом додавання додаткових стовпців, тригерів, функцій, збережених процедур тощо; фактично, розробка нової моделі надання темпоральних даних;

– реалізація підтримки на рівні додатку.

2. Постановка задачі

Метою роботи є дослідження методів підтримки темпоральності в реляційних БД шляхом їх розробки та практичної реалізації, проведення серії експериментів та формування рекомендації щодо їх ефективного використання.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз проблемної області побудови темпоральних інформаційних систем та методів підтримки темпоральності в реляційних БД;

– розробити нову модель та методи підтримки темпоральності в реляційних БД;

– спроектувати та реалізувати програмні рішення для проведення дослідження розроблених методів;

– спланувати та провести експериментальне дослідження розроблених методів та розробити відповідні рекомендації.

3. Математична модель надання темпоральних даних

Один з підходів, що досліджується, передбачає розробку, фактично, нової моделі надання темпоральних даних на базі реляційної моделі.

Отже, для забезпечення роботи РСУБД з темпоральними даними необхідна реалізація трьох основних концептів:

– темпоральних типів даних;

– видів часу;

– темпоральних тверджень.

В ході аналізу [5] виділено 3 темпоральні типи даних, що зберігають час:

– момент часу – точка на часовій осі (наприклад, CURRENT_DATE, 24 вересня 9 година ранку); цей тип даних існує і в звичайній моделі даних: майже кожна РСУБД має оператори роботи з ним, для його зберігання достатньо одного стовпця (див. рис. 1);

– інтервал – це довжина часу, тривалість відома, кінець і початок – ні;

– період – це протяжність часу, тривалість якого відома, початок і кінець визначаються двома моментами часу; період може бути закритим (включені обидва моменти часу), відкрито-закритим (один момент часу виключений), відкритим (обидва моменти часу виключені); саме цей тип даних створює багато труднощів у роботі з темпоральними моделями даних, необхідно підтримувати операції над множинами для часу (перетинання, об'єднання, різницю), обмеження цілісності, адекватність моделі даних [5].

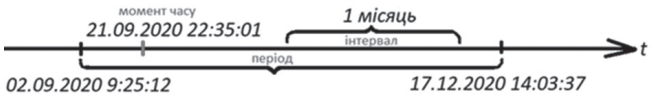


Рис. 1. Темпоральні типи даних

В ході аналізу [5] виділено 3 основних типи часу:

– час, визначений користувачем (неінтерпретований час);

– дійсний час (час, у який факт є правильним у модельованій реальності);

– транзакційний час (коли факт був записаний у БД).

В ході аналізу [5] виділено 3 базових твердження з часовим орієнтуванням:

– поточне: зараз;

– послідовне: у кожний момент часу;

– без послідовності: ігнорування часу.

Отже, три набори з трьох понять і створюють основу зберігання та управління темпоральними даними. Оскільки ці поняття є чужими для мови запитів SQL, часом виникають різні невідповідності та заміни, які кожен постачальник СУБД створив для рішення проблем темпоральності даних. Тому для реалізації темпоральних БД (ТБД) ці достатньо чіткі поняття повинні бути перетворені у громіздкі вирази та структури у SQL та схемах РБД.

Оскільки час є багатовимірним [4], то моделі даних можуть підтримувати жодного, один, два або більше вимірів:

– модель-знімок: не підтримує жодного з вимірів;

– модель з дійсним часом: підтримує лише дійсний час;

– модель з транзакційним часом: підтримує лише транзакційний час;

– бітемпоральна модель: підтримує дійсний та транзакційний час.

Мітки транзакційного часу надають інформацію про час зміни даних, виправлення помилок, а мітки дійсного – про зміну параметрів модельованого світу. Отже, дійсний і транзакційний час є ортогональними (див. рис. 2). Тому, залежно від предметної галузі та поставленої мети може бути достатньо реалізації або одного з цих типів, або двох відразу.

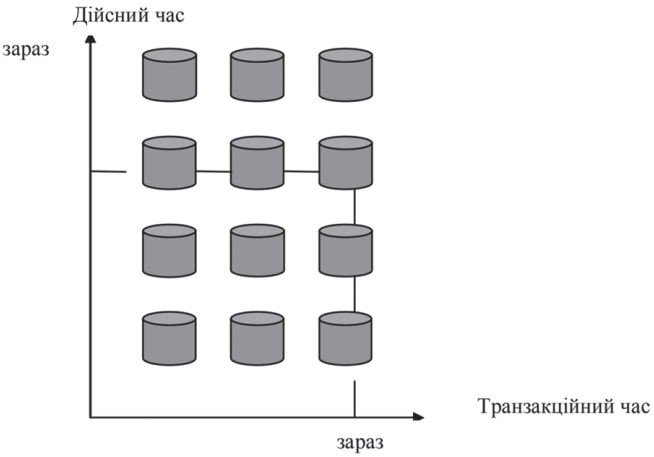


Рис. 2. Взаємовідношення між дійсним та транзакційним часом

Перейдемо до формального опису моделі темпоральних даних. На базі традиційного уявлення про модель даних [6], що складається з дозволеної організації даних, обмежень цілісності та множини операцій на об'єктами, розширена темпоральна модель (РТМ) може бути представлена, як

$$MT = (DT, OT, CT),$$

де DT – дані, OT – операції, CT – обмеження цілісності; але всі компоненти залежать від часу.

Для додання такої темпоральності була використана відкрита модель з абстрактним ідентифікатором об'єкта (Object Abstract Identify, AOID) [6], яка дозволяє будь-який об'єкт уявити у вигляді реляційного відношення. Життєвий відрізок об'єкта будемо описувати через життєві відрізки всіх його властивостей, визначених у різних відношення, що мають часові атрибути Tstart і Tend, та визначають відповідно час початку і закінчення життєвого відрізка.

Отже, об'єкт $O \in DT$ РТМ буде характеризуватися унікальним абстрактним ідентифікатором OID і набором властивостей A і може бути представлений у вигляді:

$$O = (OID, A).$$

У свою чергу набір властивостей A можна розділити на дві підмножини: множина статичних властивостей A^s, що не змінюється з часом, та множина динамічних властивостей A^d, що змінюються з часом, при цьому:

$$A = A^s \cup A^d,\ A^s \cap A^d = \varnothing.$$

Загальний вигляд РТМ представлено у вигляді ER-діаграми (див. рис. 3).

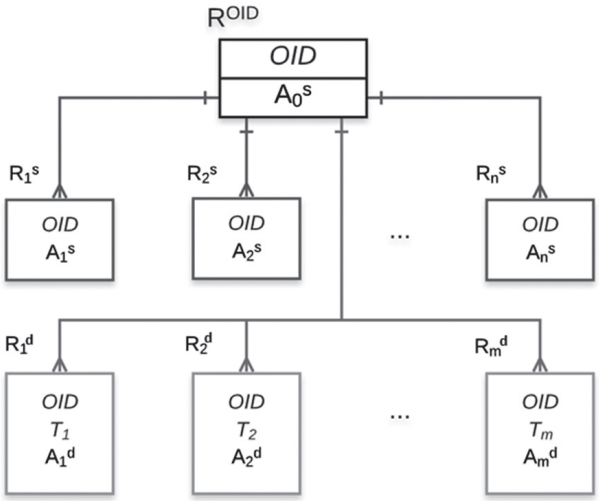


Рис. 3. ER-діаграма РТМ даних

Оскільки значення атрибутів у РБД повинні бути атомарними, то для представлення статичних та динамічних атрибутів одного реляційного відношення недостатньо. Подамо об'єкт O у вигляді сукупності взаємопов'язаних реляційних відношень:

$$O = \left(R^{OID},\ R_1^s, R_2^s, \ldots R_n^s, R_1^d, R_2^d, \ldots R_m^d\right),$$

де $R^{OID} = R^{OID}\left(OID, A_0^s\right)$ – батьківське відношення, що описує абстрактний ідентифікатор об'єкта OID та включає множину статичних атомарних атрибутів об'єкта A_0^s; отже, OID – ключ відношення R^{OID};

$R_1^s, R_2^s, \supset R_n^s$ – дочірні відношення, що описують статичні властивості A^s об'єкта;

$R_1^d, R_2^d, \supset R_m^d$ – дочірні відношення, що описують дискретно змінні у часі динамічні атрибути A^d об'єкта, та мають схеми виду:

$$R_i^d = R_i^d\left(OID, A_i^d, T_i\right),\ i = 1,2,\ldots,m,$$

де Ti – вектор атрибутів часу, що є в залежності від темпоральної форми або дійсним (VT, Valid Time), або транзакційним (TT, Transaction Time), або обома вимірами відразу, $T_i \subseteq T$, $T = \left\{t_{start}^{VT}, t_{start}^{VT}, t_{start}^{TT}, t_{end}^{TT}\right\}$ – множина атрибутів часових вимірів.

В подальшому підтримка темпоральності у моделі потребує визначення таких складових, як темпоральний ідентифікатор об'єкта OID, темпоральна алгебра OT та темпоральні обмеження цілісності CT.

Темпоральний ідентифікатор об'єкта OID може використовуватися в якості як первинного, так і зовнішнього ключів. В якості первинного ключа цей ідентифікатор однозначно може визначити стан будь-якого об'єкту у будь-який момент часу. У темпоральних моделях даних первинний та зовнішній ключі залежать від часу, адже час входить до їх складу. Додаткові обмеження цілісності CT, що накладає темпоральна модель даних, можуть бути реалізовані у РСУБД за допомогою вбудованих можливостей, таких як тригери, функції, збережені процедури.

Операції з темпоральними даними OT можуть бути реалізовані за допомогою реляційних операцій та забезпечуються у тій чи інший мірі вбудованою підтримкою РСУБД операцій з темпоральними даними (темпоральні предикати). Множина операцій має більше розбіжностей з точки зору реалізації у конкретній РСУБД, ніж, наприклад, структура даних.

Проектування запропонованої РТМ буде складатися з наступних етапів:

– концептуальне проектування з використанням моделі ER (часові аспекти ігноруються, основна задача – фіксація поточної реальності);

– логічне проектування з використанням реляційної моделі (нечасова схема ER відображається у нечасову реляційну схему, набір таблиць);

– фізичне проектування для забезпечення адекватних показників (застосовуються часові анотації, модифікуючи логічну схему). Алгоритм переходу до фізичної схеми – це вже тема окремої статті.

4. Планування експериментального дослідження

Для максимального занурення у принципи темпоральності прийнято рішення досліджувати саме бітемпоральну модель даних, щоб покрити і дійсний, і транзакційний час. Під час аналізу було виявлено, що в жодній РСУБД неможливо створити повноцінну бітемпоральну підтримку за допомогою лише вбудованої підтримки (але для окремих випадків достатньо і вбудованої), тому буде досліджуватися реалізація підтримки на рівні БД (шляхом додавання додаткових стовпців, тригерів і т.п.) (надалі, підхід «на рівні БД») та на рівні додатку (далі, підхід «на рівні додатку»).

З точки зору організації структур даних підходи не відрізняються, різниця лише у реалізації темпоральних операцій та обмежень. «На рівні БД» вони реалізуються на сервері у вигляді тригерів, функцій, збережених процедур. «На рівні додатку» – у вигляді класів, методів, тощо, залежно від обраної технології.

Тому за основу організації структур даних для обох підходів взята розроблена РТМ.

На базі РТМ для предметної області медицини була спроектована схема БД з підтримкою темпоральності (див. рис. 4). На рисунку відображений лише дійсний час (щоб забезпечити прозорість); для підтримки транзакційного часу використовуються таблиці-копії з постфіксом History усіх

темпоральних таблиць з двома додатковими стовпцями SysStartTime, SysEndTime (атрибути $t_{start}^{TT}, t_{end}^{TT}$ з математичної моделі).

Для експерименту БД була наповнена різними об'ємами даних. Дані для таблиць Drug, Disease, Procedure взяті із ресурсу https://www.cms.gov/ (вебсайт федерального уряду, керований американськими центрами Medicare & Medicaid Services), для інших таблиць дані згенеровано у різному обсязі за допомогою скриптів, що заповнюють таблиці випадковими даними.

Для обох підходів були реалізовані операції вставки, редагування, видалення та вибірки даних з урахуванням темпоральної моделі.

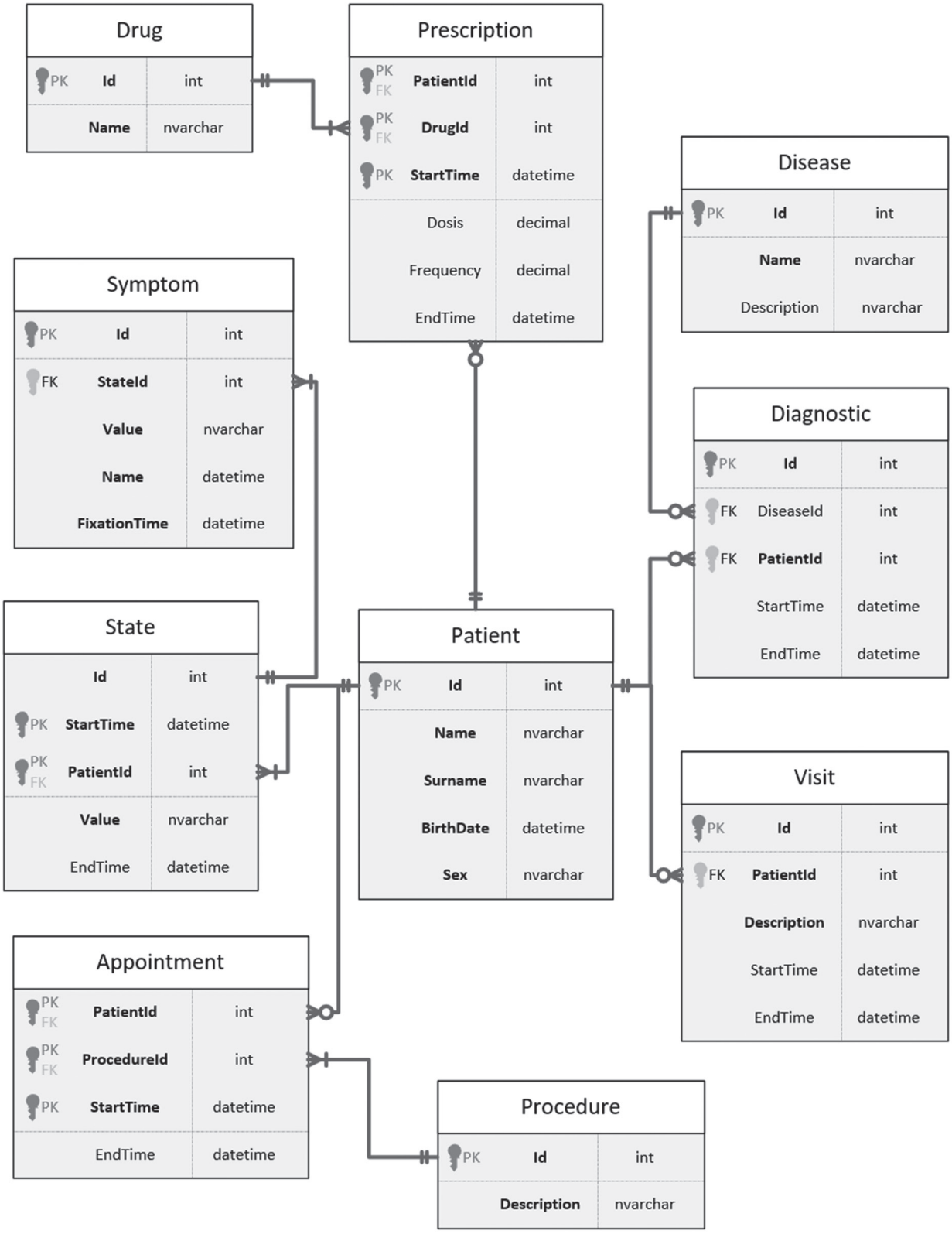


Рис. 4. Фізична схема БД з дійсним часом

Кожна вставка у бітемпоральну таблицю займає одну операцію з проставленням значень у двох колонках – дійсного часу, з якого факт починає бути істинним (атрибут t^{VT}_{start}), та транзакційного часу, у який цей факт був записаний у БД (атрибут t^{TT}_{start}). Факту, що змінюється з часом, відповідають атрибути $R^d_1, R^d_2, \supset R^d_m$ з моделі.

Кожне оновлення запису у бітемпоральній таблиці супроводжується поновленням попереднього рядка, який був істинним до операції UPDATE (проставлення дати закінчення істинності факту), та вставкою нового з проставленням дати початку істинності нового факту.

Кожне видалення з бітемпоральної таблиці полягає в проставленні дати закінчення істинності факту.

Для будь-якої операції, що змінює стан БД, перевіряються темпоральні обмеження цілісності.

Будь-яка вибірка з БД повертає дійсний стан бази даних на будь-який заданий момент часу.

Оскільки вбудованих функцій роботи з періодами немає, їх було реалізовано самостійно на рівні БД та додатку. Це такі часто використовувані функції у темпоральній логіці, як перетин періодів, включення моменту часу в період та інші. Вони є досить універсальними та не прив'язаними до конкретної предметної області, тому їх можна використовувати повторно.

Було прийнято рішення розширити експеримент та виконати його не тільки для двох обраних підходів, але й на двох СУБД, щоб отримати більш розширені висновки. Під час вибору РСУБД для проведення дослідження враховувались такі фактори, як рівень вбудованої підтримки (періоди, темпоральні предикати, операції), популярність СУБД, кількість існуючих досліджень щодо темпоральності. У результаті були обрані РСУБД Oracle та MS SQL Server.

Отже, для експериментів з MS SQL Server:

– для підходу «на рівні БД» бітемпоральну підтримку реалізовано самостійно з використанням тригерів INSTEAD OF INSERT / UPDATE / DELETE та перевіркою обмежень у них; рішення використання вбудованої в MS SQL Server підтримки транзакційного часу (відстеження зміни у таблиці, перенос недійсні записи у таблицю StateHistory) було відхилено, тому що для таких таблиць не дозволяється створювати тригери;

– для підходу «на рівні додатку» використано вбудовані можливості MS SQL Server щодо підтримки транзакційного часу, адже тригери вже непотрібні, усі перевірки цілісності йдуть на рівні додатку.

Для Oracle були прийняті наступні рішення:

– для підходу «на рівні БД» бітемпоральна підтримка реалізована самостійно з використанням тригерів: тригери BEFORE створено для перевірки обмежень, а тригери AFTER – для перенесення неактуальних даних в таблицю History; для підтримки періодів використано інструкцію *period for* при створенні таблиць з періодами (це вбудована підтримка обмежень цілісності щодо перетину періодів);

– для підходу «на рівні додатку» усі перевірки цілісності реалізовано на рівні додатку.

У ході експерименту замірялися такі характеристики, як час для SELECT/INSERT/UPDATE/DELETE операцій у темпоральних таблицях з різною кількістю записів та об'єм зайнятого дискового простору.

5. Результати проведених експериментів

Отже, під час дослідження були проведені наступні серії експериментів:

1) запити типу

SELECT/INSERT/UPDATE/DELETE

для MS SQL Server з підтримкою темпоральності на рівні БД;

2) запити типу

SELECT/INSERT/UPDATE/DELETE

для Oracle з підтримкою темпоральності на рівні БД;

3) запити типу

SELECT/INSERT/UPDATE/DELETE

для MS SQL Server з підтримкою темпоральності на рівні додатку;

4) запити типу

SELECT/INSERT/UPDATE/DELETE

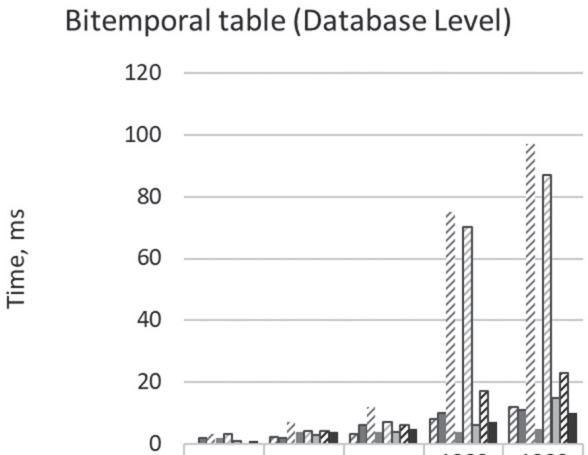
для Oracle з підтримкою темпоральності на рівні додатку.

Кожна серія експериментів проводилася на таблицях з 0, 100, 1000, 10000 та 100000 записами.

Результати експерименту перших двох серій з підтримкою на рівні БД зображені на рис. 5. Під графіком надано чисельний еквівалент у мілісекундах щодо кожного виду запиту та легенда кольорів.

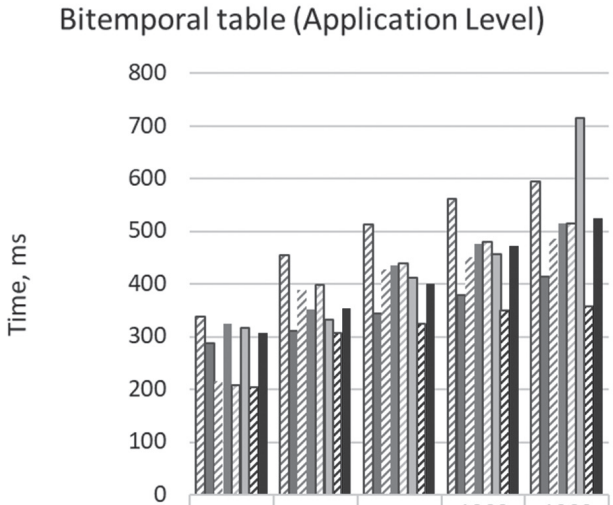
За графіками видно, що MS SQL суттєво програє Oracle на великих обсягах БД. Такі результати замовлено тим, що для перевірки цілісності періодів Oracle має оператор period for при створенні таблиць, тому у тригерах менша кількість перевірок, у MS SQL всі перевірки реалізуються самостійно за допомогою тригерів, тому INSERT / UPDATE / DELETE запити займають більше часу.

Результати серій експериментів з підтримкою на рівні додатку наведено на рис. 6. Як бачимо, є залежність від досліджуваних операцій та однозначного переможця немає. Звісно, що вся робота на рівні додатку збільшує час виконання запитів. Але для MS SQL Server ми використовуємо вбудовану підтримку транзакційного часу, завдяки чому INSERT/UPDATE/ DELETE операції виконуються швидше, ніж в Oracle. Бо в Oracle реалізуємо підтримку транзакційного часу на рівні додатку, що є повільнішим.



	0	100	1000	10000	100000
select ms sql	0	2	3	8	12
select oracle	2	2	6	10	11
insert ms sql	3	7	12	75	97
insert oracle	2	4	4	4	5
update ms sql	3	4	7	70	87
update oracle	1	3	4	6	15
delete ms sql	0	4	6	17	23
delete oracle	1	4	5	7	10

Рис. 5. Результати експериментів з Oracle та MS SQL Server для підходу «на рівні БД»



	0	100	1000	10000	100000
select ms sql	337	454	512	561	594
select oracle	287	312	345	379	414
insert ms sql	215	389	427	451	486
insert oracle	324	351	435	476	516
update ms sql	207	397	438	479	514
update oracle	316	332	412	457	715
delete ms sql	203	307	324	349	357
delete oracle	308	354	401	472	524

Рис. 6. Результати експериментів з Oracle та MS SQL Server для підходу «на рівні додатку»

Результати експериментів також було проаналізовано на базі параметру об'єму диску, що займає БД. І для MS SQL Server, і для Oracle кількість займаного місця з точки зору кількості збережених рядків однакова. Однак, таблиці з підтримкою темпоральності займають у рази більше місця, ніж звичайні, тому актуальним стає питання вакуумування. Вакуумування, що зменшує розмір History-таблиць шляхом видалення менш бажаних рядків, тим самим збільшуючи простір і ефективність роботи з таблицею, в темпоральних БД необхідно проводити обов'язково, Оскільки вбудованого функціоналу для цього немає ні в MS SQL Server, ні в Oracle, то воно реалізується самостійно.

6. Рекомендації щодо вибору підходів підтримки темпоральності у реляційних СУБД

Провівши дослідження, виконавши заміри та проаналізувавши результати, можна зазначити наступні рекомендації:

– якщо створюється система, для якої важлива підтримка лише транзакційного часу (ведення логжурналу операцій), то краще обрати у MS SQL Server з вбудованою підтримкою транзакційного; тобто використовувати підхід з підтримкою на рівні БД;

– якщо крім підтримки транзакційного часу необхідні темпоральні обмеження цілісності (не лише логування, але й елементи темпоральної логіки), то можна обрати один із трьох варіантів: реалізація перевірок обмежень цілісності на рівні додатку, або реалізація обмежень цілісності на рівні БД за допомогою збережених процедур або функцій, які треба викликати з додатка перед зміною БД, або реалізація підтримки транзакційного часу «на рівні БД»;

– якщо важлива підтримка лише дійсного часу, то краще обрати Oracle, адже Oracle має інструкцію *period for*, яка реалізує темпоральну перевірку цілісності;

– якщо треба реалізовувати підтримку і транзакційного, і дійсного часу у повній мірі, то її треба реалізовувати повністю самостійно на рівні БД що в Oracle, що в MS SQL Server, тому що:

а) у MS SQL Server неможливо створити тригери на таблицях з вбудованою підтримкою транзакційного часу, а це означає, що неможливо реалізувати перевірку темпоральної цілісності на рівні БД, що є дуже важливим при роботі з темпоральними моделями даних. Вбудованої підтримки дійсного часу зовсім немає;

б) в Oracle немає підтримки транзакційного часу, а підтримка дійсного часу теж не повноцінна;

– якщо прийнято рішення реалізувати повноцінну бітемпоральну підтримку самостійно у будь-якій СУБД, слід прийняти рішення залежно від архітектури створюваної системи, чи можлива сильна

прив'язка системи до певної СУБД, або потрібна можливість легкої зміни СУБД. Це треба вирішити на початковому етапі, тому що логіка на рівні БД повинна буде повністю переписана при використанні іншої СУБД. Якщо ж підтримка темпоральності буде реалізована на рівні додатку, то перейти на іншу СУБД буде набагато легше, але швидкість роботи буде меншою;

– у разі реалізації підтримки бітемпоральної моделі на рівні БД слід дотримуватися таких рекомендацій:

а) проектування БД здійснюється поетапно, як описується у пункті 4;

б) описується набір правил та обмежень, які повинні бути враховані у темпоральній логіці проектованої системи;

в) формується бібліотека функцій або збережених процедур, які реалізують загальну темпоральну логіку (працюють з періодами, перевіряють обмеження тощо). Надалі їх можна використовувати у тригерах для перевірки цілісності та навіть у інших системах;

г) поступово реалізується підтримка транзакційного і дійсного часу, потім додається перевірка обмежень темпоральної цілісності згідно з бізнес правилами та особливостями модельованої предметної області;

д) обирається та реалізується стратегія вакуумування на History-таблиці.

7. Висновки та перспективи

В роботі були досліджені основні аспекти підтримки темпоральності в РСУБД. Було запропоновано математичну модель представлення темпоральних даних з урахуванням дійсного та транзакційного часу.

Під час реалізації темпоральної підтримки були реалізовані загальні функції у СУБД Oracle та MS SQL Server, що можуть бути використані при роботі з будь-якими темпоральними моделями даних.

Результати експериментального дослідження дозволили сформулювати рекомендації щодо вибору підходів до роботи з темпоральними моделями даних у РСУБД. Отримані рекомендації можуть бути використані при проектуванні систем з підтримкою темпоральності.

Список літератури:

[1] Chalyi S., Leshchynskyi V. Temporal modeling of user preferences in recommender system // CEUR Workshop Proceedings. 2020. 2711. P. 518-528.

[2] Порай Д. С., Соловйов А. В., Корольков Г. В. Реализация концепции темпоральной базы данных средствами реляционной СУБД, – Едиториал, 2013. — с. 92-109.

[3] Григорович В.Г. Огляд технологій темпоральних баз даних / В.Г. Григорович, О.Ю. Косовська, О.М. Пігур-Пастернак, А.Ю. Шілінг // Вісник Національного університету «Львівська політехніка». – 2011.

[4] Krishna Kulkarni, Jan-Eike Michels. Temporal features in SQL:2011, — ACM SIGMOD Record, 2012, 34 p.

[5] Snodgrass R. Developing Time-Oriented Database Applications in SQL/Snodgrass R. — Morgan Kaufmann Publishers, 2000. — 504p.

[6] Date C.J., Hugh Darwen, Nikos A. Lorentzos. Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and, Relation Theory to the Problem of Temporal Database Management/ Date C.J., Hugh Darwen, Nikos A. Lorentzos. – Morgan Kaufmann, December, 2002. – 480p.

Надійшла до редколегії 26.03.2021