সুবিন ডট কম
তথ্যপ্রযুক্তি বিষয়ক বাংলা ব্লগ

ডাটাবেজ ট্রানজেকশন ও এসিড

রিলেশনাল ডাটাবেজের খুব মৌলিক একটি বিষয় হচ্ছে ডাটাবেজ ট্রানজেকশন। আর ডাটাবেজ ট্রানজেকশনের একটি গুরুত্বপূর্ণ বিষয় হচ্ছে এসিড প্রোপার্টি। আজকের লেখায় এই বিষয়ে আলোচনা করবো।

মনে করি জনি, রবিন, জামাল আর কামাল নামে চার বন্ধুর একই ব্যাংকে একাউন্ট আছে। ওই ব্যাংকের ডাটাবেজে Account নামে একটি টেবিল আছে এবং নিচে আমরা সেই Account টেবিলের ডাটা দেখতে পাচ্ছি –

Account No	Account Name	Balance
100-01	Jony	15000
100-02	Robin	5000

100-03	Kamal	10000
100-04	Jamal	1200

এবার আমরা নিচের ঘটনাগুলি লক্ষ করি –

- জনি তার একাউন্ট থেকে ১০০০ টাকা তুলে নিল (Cash Withdraw)
- একজন রবিনের একাউন্টে ৫০০ টাকা জমা দিল (Cash Deposit)
- কামাল তার একাউন্ট থেকে ২০০০ টাকা জামালের একাউন্টে পাঠাল (Fund Transfer)
- জামাল তার একাউন্টে কত টাকা আছে তা জানতে চাইল (Balance Enquiry)

উপরের প্রথম তিনটি ঘটনা Account টেবিলের Balance কলামের ডাটা পরিবর্তন করবে এবং শেষের ঘটনাটি Balance কলামের ডাটা পড়বে। আর রিলেশনাল ডাটাবেজ ম্যানেজমেন্ট সিস্টেমের (RDBMS) পরিপ্রেক্ষিতে এ ধরনের ঘটনাগুলিকে আমরা ডাটাবেজ ট্রানজেকশন (Database Transaction or Transaction) বা ট্রানজেকশন বলি। একটি ডাটাবেজ ট্রানজেকশনে এক বা একাধিক কাজ/ধাপ থাকতে পারে। যেমন, যখন কামাল তার একাউন্ট থেকে ২০০০ টাকা জামালের একাউন্টে পাঠাল তখন দুটি কাজ হবে প্রথমে কামালের একাউন্ট থেকে ২০০০ টাকা কমাতে হবে এবং তারপরে জামালের একাউন্টে ২০০০ টাকা বাড়াতে হবে। প্রতিটি ট্রানজেকশন হয় সফল ভাবে সম্পূর্ণ বা কমিট (Commit) হবে নয়তো রোলব্যাক (Rollback) বা ট্রানজেকশনের পূর্বের অবস্থায় ফেরত যাবে। একটি ট্রানজেকশন ডাটাতে যে পরিবর্তন করে সেই পরিবর্তিত ডাটা ডাটাবেজে স্থায়ী ভাবে রেখে দেয়াকে কমিট (Commit) বলে। আর রোলব্যাক (Rollback) হল ট্রানজেকশনের ফলে ডাটাতে যে পরিবর্তন হয়েছে সেগুলোকে বাদ দিয়ে ডাটাকে ট্রানজেকশন শুরুর পূর্বের অবস্থায় ফিরিয়ে নেয়া। উদাহরণস্বরূপ, আমরা আবার কামালের একাউন্ট থেকে ২০০০ টাকা জামালের একাউন্টে পাঠানর ট্রানজেকশনটি বিশ্লেষণ করি। আমরা জানি এই ট্রানজেকশনে দুটি কাজ করতে হবে। মনে করি প্রথম কাজটি সফল হল অর্থাৎ কামালের একাউন্ট থেকে ২০০০ টাকা কমানো হল, তাহলে কামালের একাউন্টে থাকবে ৮০০০ টাকা (ট্রানজেকশন শুরুর আগে কামালের একাউন্টে ১০০০০ টাকা ছিল)। কিন্তু কোনো কারণে দ্বিতীয় কাজটি মানে জামালের একাউন্টে ২০০০ টাকা বাড়ানো গেল না। সুতরাং এই ট্রানজেকশনটিকে রোলব্যাক করতে হবে এবং রোলব্যাক করার পরে কামালের একাউন্টে আবার ১০০০০ টাকা হয়ে যাবে।

ডাটাবেজ ট্রানজেকশন বা ট্রানজেকশনের নিম্নোক্ত চারটি বৈশিষ্ট্য/ধর্ম আছে –

- Atomicity (এটমিসিটি)
- Consistency (কন্সিসটেনসি)
- Isolation (আইসোলেশন)
- Durability (ডিউরাবিলিটি)

আর এই চারটি বৈশিষ্ট্যের প্রথম অক্ষর গুলো দিয়ে অর্থাৎ A, C, I এবং D নিয়ে আমরা বলি এসিড (ACID)। ডাটাবেজ ম্যানেজমেন্ট সিস্টেম নিজেই ট্রানজেকশনের এই বৈশিষ্ট্যগুলি পরিচালনা করে এবং অ্যাপ্লিকেশন ডেভেলপারদের আশ্বস্ত করে যে প্রতিটি ট্রানজেকশন সেগুলো মেনে চলবে।

আমারা এই চারটি বৈশিষ্ট্য উদাহরনের মাধ্যমে বোঝার চেষ্টা করি –

Atomicity – ডাটাবেজ ট্রানজেকশনের এই গুনটি নিশ্চিত করে যে, হয় একটি ট্রানজেকশনের সবগুলি ধাপ সফল হবে অথবা কোনটিই হবে না। যেমন, মনে করি, জনি তার বন্ধু রবিনকে ১,০০০ টাকা দিতে চায়। এখন জনির ব্যাংক একাউন্টে ১০,০০০ টাকা আছে আর রবিনের একাউন্টে ৪,০০০ টাকা আছে। তাহলে আমাদেরকে একটি ডাটাবেজ ট্রানজেকশন করতে হবে এবং এই ট্রানজেকশনে আমাদের দুটি কাজ/ধাপ সম্পন্ন করেতে হবে। প্রথমে জনির একাউন্ট থেকে ১,০০০ টাকা কেটে নিতে হবে এবং তারপরে রবিনের একাউন্টে সেই ১,০০০ টাকা যোগ করতে হবে। আমারা এই ট্রানজেকশনটিকে সফল বলব যদি দুটি কাজই সম্পূর্ণ হয়। আর ট্রানজেকশনটি সফল/কমিট (Commit) হলে জনির একাউন্টে থাকবে ৯,০০০ টাকা এবং রবিনের একাউন্টে হবে ৫,০০০ টাকা। এখন যদি কোনও কারণে জনির একাউন্ট থেকে টাকা কেটে নেওয়ার পরে তা রবিনের একাউন্টে যোগ করতে না পারি তাহলে আমাদের ট্রানজেকশনটি সফল/কমিট হবে না। আর ট্রানজেকশনটি সফল না হলে জনির ব্যাংক একাউন্টে ১০,০০০ টাকা এবং রবিনের একাউন্টে ৪,০০০ টাকাই থাকবে অর্থাৎ ট্রানজেকশনটি রোলব্যাক (Rollback) হবে। ডাটাবেজ ট্রানজেকশনের এটমিসিটি বৈশিষ্ট্যটি এই বিষয়টির নিশ্চয়তা দান করে।

Consistency – কন্সিসটেনসি এর বাংলা অর্থ সামঞ্জস্য বা সঙ্গতি অথবা মিল। রিলেশনাল ডাটাবেজ ম্যানেজমেন্ট সিস্টেমে কন্সিসটেনসি দ্বারা আমরা বুঝি যে প্রতিটি ডাটাবেজ ট্রানজেকশনকে ডাটাবেজে নির্ধারিত নিয়মের (Database Constraint) সাথে সামঞ্জস্য রেখে ডাটা পরিবর্তন বা নতুন ডাটা যোগ করতে হবে। আমারা নানাবিধ উপায়ে ডাটাবেজ ট্রানজেকশনের উপরে বাধ্যবাধকতা বা নিয়ম (Database Constraint) নির্ধারণ করতে পারি। যেমন, প্রাইমারি কি (primary key), ফরেন কি (foreign key), ট্রিগার (trigger), ইত্যাদি দ্বারা আমরা ট্রানজেকশনের উপরে বাধ্যবাধকতা বা নিয়ম আরোপ করতে পারি। মনে করি আমাদের একটি Student টেবিল আছে এবং studentId হল এই টেবিলের প্রাইমারি কি। আমরা যখন নতুন একজন স্টুডেন্টের ডাটা যোগ (data insert) করতে যাব তখন ডাটাবেজ পরীক্ষা করে দেখবে যে প্রাইমারি কি নিয়মটি মানা হচ্ছে কিনা। নিচের ছবিতে বিষয়টি দেখানো হল –

studentId	name	
101	John	এই ডাটাটি সফল ভাবে যোগ হবে কারন ডাটাবেজ পরীক্ষা করে দেখবে যে 101 দিয়ে আর কোনও স্টুডেন্ট নেই।

studentId	name	
102	Simon	এই ডাটাটিও সফল ভাবে যোগ হবে কারন ডাটাবেজ পরীক্ষা করে দেখবে যে 102 দিয়ে আর কোনও স্টুডেন্ট নেই।
101	Jack	এই ডাটা আমরা যোগ করতে পারবনা কারন ডাটাবেজ পরীক্ষা করে দেখবে যে 101 দিয়ে আগে থেকেই একজন স্টুডেন্ট আছে। অর্থাৎ এই ডাটাবেজ ট্রানজেকশনটি ডাটাবেজে নির্ধারিত প্রাইমারি কি এর নিয়ম অনুযায়ী সফল হবে না।

এভাবেই ডাটাবেজে নির্ধারিত নিয়মগুলো (Database Constraint) প্রতিটি ট্রানজেকশনের সময় পরীক্ষা করে দেখে যেন ট্রানজেকশনটি নিয়মের ব্যত্তয় না ঘটিয়ে সম্পূর্ণ হয়।

Isolation – আইসোলেশনের আভিধানিক অর্থ হল বিচ্ছিন্নতা। আর এই বৈশিষ্ট্যটি নিশ্চিত করে যে একাধিক ট্রানজেকশন নিরাপদে এবং স্বাধীনভাবে কোনরূপ সংঘর্ষ ছাড়া একই সময়ে সম্পূর্ণ হতে পারে, কিন্তু এটা কোন ট্রানজেকশনটি আগে হবে আর কোনটি পরে হবে অর্থাৎ ক্রম (order) নিশ্চিত করে না। উদাহরণস্বরূপ, মনে করি রনির একাউন্টে ১৫,০০০ টাকা আছে। রনি তার দুই বন্ধু কামাল এবং জামাল কে যথাক্রমে ৩,০০০ ও ২,০০০ টাকার দুটি চেক দিল। কামাল এবং জামাল একসাথে ব্যাংকে গেল টাকা তোলার জন্য। তারা দুজন ব্যাংকের দুজন অপারেটরের কাছে একই সময়ে চেক দুটি জমা দিল। এখানে একই সাথে দুটি ট্রানজেকশন হবে, কিন্তু যেহেতু একই একাউন্ট থেকে টাকা তোলা হবে তাই যে কোনও একটি ট্রানজেকশন আগে হবে এবং অন্যটিকে অপেক্ষা করতে হবে। ধরে নেই জামালের ট্রানজেকশনটি আগে শুরু হল তাই কামালের ট্রানজেকশনটি অপেক্ষা করবে। জামালের ট্রানজেকশনটি সম্পূর্ণ হলে কামালের ট্রানজেকশনটি শুরু হবে। অর্থাৎ জামালের ট্রানজেকশনটি যখন শুরু হবে তখন রনির একাউন্টে আছে ১৫,০০০ টাকা আর ট্রানজেকশনটি শেষ হবার পরে রনির একাউন্টে ১৩,০০০ টাকা থাকবে। আর কামালের ট্রানজেকশনটি যখন শুরু হবে তখন রনির একাউন্টে আছে ১৩,০০০ টাকা আর ট্রানজেকশনটি শেষ হবার পরে রনির একাউন্টে ১০,০০০ টাকা থাকবে। যেহেতু দুটি ট্রানজেকশনই একই ডাটার (রনির একাউন্ট) উপর নির্ভরশীল তাই একটিকে অন্যটি শেষ হবার জন্য অপেক্ষা করতে হচ্ছে। যদি এভাবে না হত তাহলে যে ডাটার উপর ট্রানজেকশনগুলি নির্ভরশীল সেই ডাটা একটা সামঞ্জস্যহীন (inconsistent) অবস্থায় চলে যাবে। আর ডাটা যেন কোনও ভাবেই সামঞ্জস্যহীন না হয় সে জন্যই ট্রানজেকশন আইসোলেশন প্রয়োজন।

ডাটাবেজে চারটি আইসোলেশন লেভেল আছে –

1. Read Uncommitted
2. Read Committed
3. Repeatable Read
4. Serializable

Read Uncommitted হল আইসোলেশনের সর্বনিম্ন লেভেল আর Serializable হচ্ছে সর্বোচ্চ লেভেল। এই আইসোলেশন লেভেলগুলির কিছু সমস্যা আছে যথা, Dirty Reads, Non Repeatable Reads এবং Phantom। নিচে এগুলোর বর্ণনা দেয়া হল –

Dirty Read – একটি ট্রানজেকশন অন্য ট্রানজেকশনের দ্বারা পরিবর্তিত ডাটা যা কমিট (commit) হয়নি সেগুলো পড়তে পারাকেই ডারটি রিড বলে। উদাহরণ স্বরূপ, মনে করি একজন ক্রেতা একটি কেনাকাটার সাইট থেকে কোনও একটি পণ্য ২৮০ টি কিনতে চাইল, এখন তার জন্য ট্রানজেকশন A শুরু হল। ট্রানজেকশন A প্রথমে দেখবে যে ঐ পণ্যের ২৮০ টি স্টকে আছে কিনা। ধরে নেই সাইটের ডাটাবেজে Product_Inventory নামে একটি টেবিল আছে যাতে পণ্যের পরিমান আছে। তো ট্রানজেকশন A সেই Product_Inventory টেবিল থেকে পেল যে তার ইউজার যে পণ্যটি কিনতে চায় তা ৫০০ টি আছে। সুতরাং ট্রানজেকশন A এই অর্ডারটিকে কনফার্ম করল এবং Product_Inventory টেবিলে ঐ পণ্যটির পরিমান ৫০০ থেকে কমিয়ে ২২০ করে দিল। কিন্তু ট্রানজেকশন A এখনও কমিট হয়নি। একই সময়ে আরও একজন ঐ একই সাইট থেকে ঠিক ঐ পণ্যটি ৪০০ টি কিনতে চাইল। ধরে নেই পরের ক্রেতার জন্য ট্রানজেকশন B শুরু হল। ট্রানজেকশন B দেখল যে ঐ পণ্যটি মাত্র ২২০ টি আছে, তাই সে ক্রেতাকে জানাল যে তার অর্ডারটি নেয়া যাচ্ছে না। এরই মধ্যে আবার ট্রানজেকশন A এর যে ক্রেতা সে অর্ডারটি বাতিল করে দিল, তার ফলে ট্রানজেকশন A রোলব্যাক হয়ে গেল। অর্থাৎ Product_Inventory টেবিলে ঐ পণ্যটির পরিমান আবার ৫০০ হয়ে গেল। তার মানে এখানে ট্রানজেকশন B এমন একটি ডাটা পেয়েছিল যা আসলে কমিট হয়নি আর এটাকেই Dirty Read বলে।

Non Repeatable Read – একটি ট্রানজেকশন যদি একই ডাটা দুবার পড়ে আর দুবার দুটি আলাদা ভ্যালু পায় তাকে Non Repeatable Read বলে। যেমন, মনে করি রবিন একটি কেনাকাটার সাইট থেকে কোনও একটি পণ্য ৩০০ টি কিনতে চাইল, এখন তার জন্য ট্রানজেকশন A শুরু হল। ট্রানজেকশন A প্রথমে দেখবে যে ঐ পণ্যের ৩০০ টি স্টকে আছে কিনা। ধরে নেই সাইটের ডাটাবেজে Product_Inventory নামে একটি টেবিল আছে যাতে পণ্যের পরিমান আছে। তো ট্রানজেকশন A সেই Product_Inventory টেবিল থেকে পেল যে রবিন যে পণ্যটি কিনতে চায় তা ৫০০ টি আছে। একই সময়ে জামাল ঐ একই সাইট থেকে ঠিক ঐ পণ্যটি ২৫০ টি কিনতে চাইল। ধরে নেই জামালের জন্য ট্রানজেকশন B শুরু হল এবং এই ট্রানজেকশনটি Product_Inventory টেবিল থেকে পেল যে ঐ পণ্যের ৫০০ টি স্টকে আছে। এদিকে রবিন তার অর্ডারটিকে কনফার্ম করল তার ফলে ট্রানজেকশন A পণ্যটির পরিমান ৫০০ থাকে কমিয়ে ২০০ করে দিল এবং ট্রানজেকশন A কমিট হয়ে গেল। ওদিকে জামাল তার অর্ডারে একটু পরিবর্তন করল, সে ঐ পণ্যটি বাদ দিয়ে অন্য একটি পণ্য ২৫০ টি কিনতে চাইল। ফলে

ট্রানজেকশন B আবার Product_Inventory টেবিল থেকে খুঁজে পেল যে এই পরে অর্ডার দেয়া পণ্যটি মাত্র ১০০ টি রয়েছে। তাই ট্রানজেকশন B জামালকে জানাল যে পরে অর্ডার দেয়া পণ্যটি স্টকে ২৫০ টি নেই, জামাল আবার প্রথমে অর্ডার দেয়া পণ্যটি নিতে চাইল। এবার ট্রানজেকশন B Product_Inventory টেবিল থেকে দেখল যে এই পণ্যের আর ২০০ টি অবশিষ্ট আছে এবং অর্ডারটি নেয়া যাচ্ছে না। অর্থাৎ ট্রানজেকশন B একই ডাটা দুবার পড়ে দুটি ভিন্ন ভ্যালু (পণ্যের দুটি ভিন্ন পরিমান) পেল। ট্রানজেকশন B যে সমস্যাটির মুখে পড়েছে তাকে Non Repeatable Read বলে।

Phantom – মনে করি কেনাকাটার সাইটের ডাটাবেজে Order নামে একটি টেবিল আছে এবং এই টেবিলে সব অর্ডারের ডাটা আছে। এখন একজন জানতে চাইল মোট কতগুলো অর্ডার হয়েছে। তো একটি ট্রানজেকশন A শুরু হল যা কিনা Order টেবিল থেকে মোট কতটি অর্ডার হয়েছে তা বের করবে। এদিকে একই সময়ে অন্য একটি ট্রানজেকশন B নতুন একটি ডাটা Order টেবিলে যোগ করল। এখন যদি ট্রানজেকশন A আবার Order টেবিল থেকে মোট কতটি অর্ডার হয়েছে তা বের করে তাহলে ভিন্ন ভ্যালু পাবে। এই ধরনের ঘটনাকে Phantom বলে। Non Repeatable Read এর সাথে Phantom এর পার্থক্য হচ্ছে এখানে ডাটা ভিন্ন হচ্ছে নতুন ডাটা যোগ করার ফলে বা ডাটা মুছে ফেলার কারণে (insert or delete)।

নিচের টেবিলে কোন আইসোলেশন লেভেলে কোন অসুবিধা গুলো হয় বা হয় না তা দেখানো হল –

Isolation Level	**Dirty Read**	**Non Repeatable Read**	**Phantom**
Read Uncommitted	হয়	হয়	হয়
Read Committed	হয় না	হয়	হয়
Repeatable Read	হয় না	হয় না	হয়
Serializable	হয় না	হয় না	হয় না

Durability – ডিউরাবিলিটি শব্দের মানে স্থায়িত্ব। ডাটাবেজ ট্রানজেকশনের এই ধর্মটি নিশ্চিত করে যে, যখন একটি ট্রানজেকশন সফল হয় তখন তার ফলে যে ডাটা পরিবর্তন হয় তা যেন স্থায়ীভাবে ডাটাবেজে থেকে যায়। এর অর্থ হল, সব ধরনের অঘটন/দুর্ঘটনা সত্ত্বেও (System Failure/System Crash) বা সিস্টেম রিস্টার্ট (System Restart) হলেও সফল ট্রানজেকশনের দ্বারা ডাটাতে যে পরিবর্তন হয়ছে তা স্থায়ী ভাবে ডাটাবেজে সংরক্ষিত থাকবে। যেমন, মনে করি আমাদের একটি বাসের টিকেট বুকিং সিস্টেম আছে। এই বুকিং সিস্টেমে রবিন একটি টিকেট বুকিং দিল এবং তার বুকিংটি সফল হল, আর এরপরেই এই বুকিং সিস্টেমটি বৈদ্যুতিক গোলযোগের (Power Failure) কারণে (System Crash) ক্র্যাশ করল।

কিন্তু যেহেতু রবিনের বুকিংটি সফল হয়েছিল তাই বুকিং সিস্টেমটি পুনরায় চালু হবার পরে তার করা বুকিংটি ডাটাবেজে পাওয়া যাবে।

আশা করি লেখাটি ডাটাবেজ ট্রানজেকশন ও এসিড সম্বন্ধে বুঝতে সাহায্য করবে।

লেখকঃ মোঃ শফিউজ্জামান রাজিব, ডাটাবেজ ও বিগ ডাটা প্রফেশনাল।

Facebook Comments

10 Comments

Sort by Top ▼

Add a comment...

Monorongon Ray Sattajit

very good

Like · Reply · Mark as spam · 1w

ফিনিক্স এর রিহান

পড়ে অনেক ভাল লাগলো। অনেক সহজবধ্য করে লেখা।

Like · Reply · Mark as spam · 51w

Jobair Ahmed

valo lagce

Like · Reply · Mark as spam · 1y

Engr Md Mahedi Hasan

It's a very good written about ACID in RDBMS. Thanks for sharing among with us..

Like · Reply · Mark as spam · 2y

নিরব হৃদয়

thanks, eirokom aro post asha korche.
ডাটাবেজ নিয়ে কাজ শিখতে চাইলে এই পেইজের সাথে থাকুন
https://www.facebook.com/oracle.bangla.shohag/
website: http://oraclebangla.com/

Like · Reply · Mark as spam · 2y · Edited

Emdadul Huq

অসাধারণ। ডাটাবেজ নিয়ে আরো আর্টিকেল আশা করছি।

Like · Reply · Mark as spam · 👍 1 · 2y · Edited

Sheikh Abdullah

thanks

Like · Reply · Mark as spam · 2y

Jahid Hasan

পর পর ওনার দুই পোষ্ট। বেশ ভালোই লাগলো। একটা বই হয়তো বাহির করতে পারে

Like · Reply · Mark as spam · 2y