پایگاه داده پیشرفته

محمد داورپناه جزی

ترم دوم ۹۴-۹۳

گروه مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی فولاد

روش مدیریت درس

- **تکالیف درسی** ٪10
- **مقاله + ارائه در کلاس** ٪15
- **امتحان میان‌ترم** ٪30 **+ جایزه**
- **امتحان پایان‌ترم** ٪45 **+ جایزه**
- **روز حساب دارای نمرهٔ اضافی**
- **اسلایدهای درس روی سایت** davarpanahjazi.iut.ac.ir/fa
- **مرجع اصلی درس (متن روی سایت):**

An Introduction to Database Systems Volume 2

C J Date

- **سایرمراجع درس: دریای بیکران اینترنت**

فهرست مطالب

- **فصل اول ترمیم**
- **فصل دوم یکپارچگی**
- **فصل سوم توازی**
- **فصل چهارم امنیت**
- **فصل پنجم مدلهای داده**
- **فصل ششم مدل رابطه ای توسعه یافته**
- **فصل هفتم پایگاه داده توزیع شده**
- **فصل هشتم ماشینهای پایگاه داده ای**

تاکید درس روی مطالب تئوری موضوعات بالا و انجام قسمتهای کاربردی به عنوان تکلیف درسی

فصل اول: ترمیم (recovery), مقدمه

اصل کلی: عدم کارکرد ..ا% سیستمهای کامپیوتری در کل و DBMS ها به طور خاص

راه حل: لزوم به کار گیری تستها و کنترلها (حتی الامکان) برای پیشگیری از شکست یا failure

وجود حجم قابل توجهی کد برای پشتیبانی سیستم در ترمیم پس از شکست اجتناب ناپذیر

سیستم R ده درصد, IMS از این هم بیشتر (مثالهایی از ترمیم در زندگی روزمره)

ترمیم: برگرداندن DB به حالت مطمئنا درستی در قبل از شکست, بعد از اثبات یا حدس خرابی

دلایل شکست: اشکال در برنامه, DBMS, OS, HW, ارتباطات, برق, اپراتور, آتش, ...

مبنای ترمیم: به کار گیری افزونگی یا redundancy در داده ها برای ایجاد امکان بازسازی مجدد پایگاه بدون از دست دادن کمترین داده, طی مراحل ساده شده زیر:

ا: کپی برداری ادواری (روزانه) از کل پایگاه و حفظ در داخل و بیرون سازمان

۲: log همه اصلاحات در قالب مقدار قبلی و فعلی

۳: برخورد با شکست, الف) خرابی کل پایگاه: اعمال اصلاحات log روی آخرین کپی

ب) توقف غیر عادی: برگرداندن اصلاحات به نقطه امن بر مبنای log

فصل اول: ترمیم (recovery)، (ادامه)

واقعیت ترمیم: لزوم انجام عملیات بسیار پیچیده و دقیق در مقایسه با مراحل بالا

تمرکز مطالب روی جنبه های مفهومی و تئوری در کل و عدم تاکید روی سیستم خاص، صرف نظر از مواردی چون بهینه سازی برای سادگی در شروع

یک روش عملی: دوپلکس کردن یعنی ایجاد دو نسخه از پایگاه روی دو محیط متفاوت

انجام کلیه اصلاحات روی هر دو محیط به طور کاملا مشابه و همزمان (تقریبا؟)

مزایا: بالا بردن قابلیت اطمینان، ارتقای کارایی! پیاده سازی در برخی محیطها

معایب: نیاز به فضای حافظه دوبرابر، لزوم استقلال کامل (تقریبا؟) دو سیستم از دید سخت افزاری (دیسک، کانالها، ...)، نرم افزاری و حالات ممکن شکست

کارکرد تحت یک CPU واحد (معمولا)، کماکان نیار به log برای برگشت به عقب

نیاز به log مجزا در برخی سیستمها برای استخراج داده های آماری و بررسی کارکرد

صرفه؟: تقسیم داده ها به ترمیم پذیر و ناپذیر، فعلا فرض بر ترمیم پذیری کل پایگاه

فصل اول: ترمیم, ۱-۲ تراکنشها

تراکنش: یک واحد کاری قابل تعریف شامل اجرای مراحل زیر

اول: **Begin transaction** بعد در صورت خاتمه با موفقیت: **Commit**

و گرنه, خاتمه ناموفق : **Rollback**

خاتمه: برای تراکنش و نه برنامه (ش ۱-۱ ص ۳), ولی معمولا یک تراکنش در هر برنامه

شرط مهم: تودرتو نبودن تراکنشها, اجرای **BT** درصورت نبود تراکنش درحال اجرا اجرای **CMT** یا **RB** درصورت وجود تراکنش درحال اجرا,

ضمنی بودن **BT** معمولا, همراه بودن **BT** با شروع برنامه و پس از هر **CMT** و **RB**

عملیات قابل ترمیم: اجرا در قالب تراکنش, وجود امکان **Redo** و **Undo** (ثبت **log**)

مثال عملی: انتقال پول بین دو حساب, ش ۱-۲ ص ۵ و دستور ص ۴

از دید کاربر یک عمل اتمی, در عمل دو تراکنش,

لزوم انجام کامل کل عملیات بدون شکست در وسط کار

فصل اول: ترمیم, ۱-۲ تراکنشها (ادامه)

سیاست همه یا هیچ: اجرای کامل تراکنش یا عدم اجرای آن به طور کلی

اجرا فقط یک بار, حتی در صورت تکرار هم نتیجه معادل یک بار اجرای واقعی

نتیجه: لزوم قابل اعتماد بودن (عدم اجرای نصفه نیمه, عدم اجرای بیش از یکبار و عدم گمشدگی) و رعایت سیاست **AoN** چه در مورد عملیات و چه در مورد پیامها (مثال . . .)

تقسیم وظایف: وجود مدیریتهای مختلف در هر **DBMS**, یکی هم مدیر ترمیم یا **RM**

مدیر ترمیم: انجام کلیه موارد مرتبط با ترمیم پایگاه, اشاره مجدد به ش ۱-۲

پیامها: اهمیت شرایط و محل ارسال پیام به کاربر, نمونه هایی ار پیام در ش ۱-۲,

ارائه پیام در صورت اجرای کامل و **RB** در حالت بروز اشکال (تقسسیم به صفر, سرریز, . . .), لزوم اعلام پیامهای خروجی بعد از خاتمه برنامه ریزی شده (مثال **ATM**), امکان **queue** کردن پیامها و حذف یا اعلام در خاتمه بر مبنای وضعیت تراکنش,

اهمیت موضوع: وقتی پیام منجر به عملی مثل تحویل پول باشد و تراکنش **RB** شود!

فصل اول: ترمیم, ۱-۲ تراکنشها, پیامها

مدیریت پیامها: توسط **Data communication (DC) manager**,

ماهیت پیام: دارای شرایط مشابه تراکنشها, لزوم استقلال پیامها در هر **DBMS** و حفظ در قالب رکورد منطقی بدون توجه به فرمت خروجی (یک نگاشت وظیفه مثلا **MFS**)

روش عمل: ایجاد دو صف ورودی و خروجی پیامها روی دیسک

پیام ورودی: نوشتن **log** و ثبت در صف ورودی, تحویل به عمل **Get**

پیام خروجی: ثبت پیام **Put** شده در صف برای تصمیم بعدی

زمان **CMT** یا **RB** برنامه ریزی شده: نوشتن **log** مربوط به صف خروجی,
ارسال واقعی پیام, حذف پیام ورودی مربوطه (پردازش شده) از صف

توقف غیرطبیعی: حذف پیامهای مرتبط از صف خروجی,
انجام **redo** یا **undo** برمبنای **log**

توجه: انجام کلیه عملیات بالا توسط **DCM** بر مبنای سیاست **AoN**

فصل اول: ترمیم, ۱-۲ تراکنشها, ساختار تراکنش

ساختار کلی تراکنش: گرفتن پیام ورودی ← انجام پردازش در DB ←ارسال پیام خروجی,
وابستگی و تداخل جزئیات مراحل سه گانه

ماهیت یک محاوره: برقراری ارتباط چندگانه در چند مرحله بین انسان و ماشین, دو روش

روش اول: انجام چندین تراکنش ریز به ترتیب بالا و **CMT** یا **RB** مستقل هر یک

روش دوم: انجام یک تراکنش مفصل شامل چندین مجموعه ورود-پردازش-خروج و نهایتا ختم با **CMT** یا **RB**

مشکل روش اول: اعمال اصلاحات بین تراکنشهای ریز و غلط شدن عملیات

مشکل روش دوم: لزوم ارسال پیامها به کاربردر طول تراکنش مفصل (خلاف تصمیم قبل)
,نیاز به اعلام لغو همه پیامها درصورت **RB** در نهایت!
مسئولیت این امر با برنامه کاربر و نه سیستم!

در این درس: فرض بر انجام روش اول و ارسال واقعی پیامها در آخر کار

فصل اول: ترمیم, ۱-۲ تراکنشها, انواع شکست

یک نمونه: توجه به ش ۱-۲ ص ۵ برای RB توسط برنامه و پیشگیری از خرابی DB

حالت کلی: عدم امکان پیش بینی شکستها در همه موارد توسط برنامه کاربر

انواع شکست: طبقه بندی شکستها بر مبنای منبع آنها

وابسته به تراکنش: حالاتی که در برنامه کاربر کنترل و پیشگیری می شود (ش ۲-۱)

مخفی در تراکنش: کنترل نشده در تراکنش مثل سرریز یا تقسیم به صفر

وابسته به سخت افزار سیستم: موثر روی تراکنشهای جاری ولی نه DB,

مثل وجود مشکل در CPU و توقف سیستم

وابسته به محیط ثبت داده: موثر روی کل DB یا بخشی از آن و تراکنشهای مرتبط,

مثل خراب شدن دیسک و عدم امکان دسترسی به آن

مورد اول بررسی شد, بررسی سه مورد بعدی در ادامه

فصل اول: ترمیم, ۱-۳ شکست تراکنش

شکست: قطع ناخواسته و غیرطبیعی (Abend) برنامه به دلایلی مانند تقسیم به صفر, رجوع به آدرس غلط و یا بروز سرریز در محاسبات

روند: Error → Fault → Failure

مدیریت: on unit خاص (on zerodivide), بررسی کلی (on error),

واگذاری به سیستم به معنی system action یا system failure

شکست: یک خطای برنامه ریزی نشده و نه یک شرط منجر به RB

جملات UDL: قابل چک کردن در چهار سطح

سطح اول: قرار دادن not found در خود دستور find

سطح دوم: به کار بردن on not found به صورت کلی (دینامیک)

سطح سوم: به کار بردن on DBerror برای همه خطاها

سطح چهارم: واگذاشتن عکس العمل به ماشین (توصیه نمی شود)

شکست تراکنش: نرسیدن به پایان قابل انتظار

فصل اول: ترمیم, ۱-۳ شکست تراکنش (ادامه)

شکست تراکنش: نرسیدن به پایان قابل انتظار یعنی CMT یا RB

عکس العمل RM: برگرداندن عملهای ناقص (قطعی/احتمالی) و لغو پیغامها, به نحوی که انگار هیچ اتفاقی نیفتاده

لغو پیغامها: حذف از لیستهای مربوطه به سادگی

لغو عملیات: نیاز به دقت! بررسی log تا نقطه BT و جایگزینی مقادیر فعلی (جدید؟) با مقادیر قبلی برای کلیه تغییرات

عمل undo: وجود سه روال ایجاد, اصلاح و حذف در هر DBMS و همچنین سه روال برای undo آنها, مورد احضار توسط RM برای برگرداندن تغییرات

Online log: قرار دادن log روی دیسک برای دسترسی مستقیم, حجیم شدن آن در DBMSهای عادی (۲۰۰ مگ/روز), دسترسی زمانبر, نیاز به log manager (مدیر ۳)

راه حل: آرشیو کردن ادواری log, یک active log و تعدادی archive log, لزوم حفظ سوابق تراکنشهای جاری روی active, احتمال RB آنها درصورت سرریز شدن ACL

فصل اول: ترمیم, ۱-۳ شکست تراکنش (ادامه)

منطق undo: امکان شکست خود عمل RB و نیاز به تکرار آن, لزوم برقراری رابطه زیر

$$\forall x \; undo(undo(\ldots(undo(x))\ldots) = undo(x)$$

لازم است عمل undo خاصیت idempotent یا تکرارپذیری داشته باشد.

تراکنشهای طولانی: یک تراکنش واحد = یک واحد کار = یک واحد قابل ترمیم

مشکل: نیاز به حجم زیاد undo و redo در موارد شکست, احتمال سرریز شدن log و ...

راه حل: تقسیم تراکنشهای طولانی به تعدادی تراکنش کوتاه و CMT بعد از هر یک

مثال: برنامه محاسبه حقوق ش ۱-۳, پردازش اسامی شروع شده با یک حرف در هر نوبت

انجام CMT بعد از خاتمه هر حرف و نوشتن یک رکورد restart , شروع با حرف بعدی در صورت شکست, ۲۶ تراکنش کوتاه به جای یک تراکنش طولانی

متراکم کردن آرشیو Log: حذف مواردی چون سوابق RB, مقادیر قبلی تراکنشهای CMT شده برای redo, حفظ آخرین مقدار از اصلاحات مکرر یک رکورد, ایجاد ترتیب مشابه با DB برای تسریع ارجاعات بعدی به log

فصل اول: ترمیم, ۱-۴ شکست سیستم

شکست سیستم: به معنی توقف ناخواسته سیستم و لزوم **restart**

از دست دادن محتویات حافظه و بافرها, کامل نشدن تراکنشهای جاری,

لزوم بر گرداندن آنها, سلامت خود **DB**

مشکل RM: چگونگی تشخیص تراکنشهای ناتمام, جستجو در **log** برای **BT**های بدون **CMT** یا **RB** عملی کند و زمانبر

یک راه حل: ایجاد **chkpnt** ادواری (هر **n** دقیقه یا نوشتن **m** رکورد **log**) به ترتیب زیر

مرحله ۱: الزام به نوشتن بافرهای **log** داخل حافظه به فایل **log**

مرحله ۲: نوشتن یک رکورد **chkpnt** روی فایل **log**

مرحله ۳: الزام به نوشتن بافرهای **DB** داخل حافظه به **DB** روی دیسک

مرحله ۴: نوشتن آدرس رکورد **chkpnt** داخل فایل **restart**

توضیح در مورد **force writing** (بافرینگ و بلوکینگ برای بالا بردن کارایی)

مراحل ۱ و ۲ برای سازگاری با پروتکل **log write-ahead**, مرحله ۳ برای نهایی سازی عملیات قبل از **chkpnt** و پیشگیری از **redo** آنها

فصل اول: ترمیم, ۱-۴ شکست سیستم (ادامه)

رکورد **chkpnt**: حاوی لیست تراکنشهای فعال در زمان **chkpnt** همچنین آدرس آخرین رکورد مرتبط با هر کدام در **log**

وظایف RM در restart: برداشتن آدرس آخرین رکورد **chkpnt** از فایل **restart**,

بازیابی رکورد مزبور از فایل **log**, حرکت به سمت آخر **log** برای تشخیص موارد **redo** و **undo** برای تبدیل وضعیت **DB** به حالت درست,

شرح الگوریتم در ادامه

حالات تراکنش: وجود پنج گروه تراکنش در زمان شکست بعد از یک **chkpnt**, ش ۱-۴

گروه ۱: تراکنش **T1** کامل شده قبل از **chkpt**, عدم نیاز به ترمیم (همه بدون دخالت کاربر)

گروه ۲: تراکنش **T2** شروع قبل از **chkpnt** تکمیل بعد از آن و قبل از شکست, لزوم **redo**

گروه ۳: تراکنش **T3** شروع قبل از **chkpnt** ولی عدم تکمیل قبل از شکست, لزوم **undo**

گروه ۴: تراکنش **T4** شروع بعد از **chkpnt** تکمیل قبل از شکست, لزوم **redo**

گروه ۵: تراکنش **T5** شروع بعد از **chkpnt** ولی عدم تکمیل قبل از شکست, لزوم **undo**

فصل اول: ترمیم, ۱-۴ شکست سیستم (ادامه)

الگوریتم recovery: لزوم انجام موارد زیر توسط RM به طور اتوماتیک

۱- ایجاد لیست undo حاوی کلیه تراکنشهای ثبت شده در رکورد chkpt

۲- ایجاد لیست خالی redo

۳-شروع از آخرین نقطه chkpnt بازیابی شده در log و حرکت به طرف آخر log

۴- ثبت تراکنشهای دارای BT به لیست undo

۵- انتقال تراکنشهای دارای CMT از لیست undo به لیست redo

۶- مشخص شدن موارد redo و undo بعد از رسیدن به خاتمه log

۷- حرکت از آخر log به عقب برای undo تراکنشهای داخل لیست undo

۸- حرکت به طرف آخر log برای redo تراکنشهای داخل لیست redo

توضیح: عدم قبول هر گونه اصلاح در طول مراحل بالا.

در نوع T2 عمل redo برای موارد بعد از chkpnt,

در نوع T3 عمل undo برای موارد قبل از chkpnt

فصل اول: ترمیم, ۱-۴ شکست سیستم (ادامه)

عمل redo: وجود سه روال ایجاد, اصلاح و حذف در هر DBMS و سه روال برای undo آنها, همچنین نیاز به سه روال برای redo آنها همه مورد احضار توسط RM

منطق redo: امکان شکست خود عمل redo و نیاز به تکرار آن, لزوم برقراری رابطه زیر

$$\forall x \; redo(redo(\ldots(redo(x))\ldots) = redo(x)$$

لازم است عمل redo نیز مشابه undo خاصیت idempotent یا تکرارپذیری داشته باشد.

پروتکل log write-ahead: مشکل استقلال دو عمل نوشتن روی log و روی DB, امکان بروز شکست بین این دو نوشتن!

لزوم نوشتن log قبل از DB برای ایمنی با رعایت دو شرط زیر:

۱- عدم ثبت رکورد روی DB فیزیکی توسط تراکنش مگر بعد از ثبت حداقل بخش مربوط به undo روی log فیزیکی

۲- عدم انجام CMT توسط تراکنش مگر بعد از ثبت هر دو بخش مربوط به undo و redo کلیه رکوردهای log آن تراکنش روی log فیزیکی

نتیجه: امکان انجام undo و یا redo موجود در log که روی DB اصلا انجام نشده! (مشکل؟)

فصل اول: ترمیم, ۱-۴ شکست سیستم (ادامه)

شروع مجدد سیستم: عمل **system start-up** شامل انواع زیر

۱- **emergency restart**: شروع مجدد بعد از شکست شامل ترمیم با استفاده از **log**, احتمال نیاز به آوردن کل **DB** از آرشیو

۲- **warm start**: شروع مجدد بعد از متوقف کردن طبیعی (**shut down**) شامل تکمیل تراکنشهای جاری و بستن **log** و **DB**, عدم نیاز به ترمیم در این نوع شروع بر مبنای **log**, آخرین رکورد **log** رکورد **chkpnt** فاقد تراکنش جاری

۳- **cold start**: شروع سیستم برای اولین بار یا پس از یک خرابی کلی,

امکان نیاز به آوردن **DB** از آرشیو و تکرار اصلاحات از آن زمان به صورت دستی! ایده آل فقط یکبار

نکته: رعایت احتیاط برای موارد ۱ و ۲ در نوشتن دونسخه فایل **restart** برای رجوع به آخرین دو **chkpnt**

پیغامها: مشابه تاثیر **undo** تراکنش روی پیامها, وجود تاثیرمشابه در **redo** تراکنش,

نیاز به **log** کردن پیام ورودی تراکنشهای **T3** و **T5** برای **redo** آنها,

عدم تکرار پیامهای خروجی **T2** و **T4** در **redo** آنها, ارسال پیامهای اولیه نهایتا

فصل اول: ترمیم, ۱-۴ شکست سیستم, اصلاح الگوریتم ترمیم

مشکل الگوریتم قبلی: انجام **redo** و **undo** به صورت کور کورانه, فرض بر نوشته شدن کلیه اصلاحات از نوع T3 و T5 روی DB فیزیکی و نوشته نشدن نوع T2 و T4 کلا

نتیجه: **undo** شدن کل گروه اول و **redo** شدن کل گروه دوم بر مبنای فرض غلط

راه حل: انجام مرحله ۳ (**الزام به نوشتن بافرهای DB**) طبق منطق مشخص در زمان لازم

ایجاد منطق برای پیشگیری از **redo** و **undo** غیر لازم به شرح زیر

۱- تخصیص شماره ردیف (LSN) به رکوردهای **log**

۲- نوشتن LSN رکورد **log** مربوط به هر بلوک مورد ثبت روی **DB** فیزیکی در آن

۳- با فرض شماره ردیف r برای رکورد R در **log** مربوط به یک اصلاح در **DB** ثبت شده در بلوک P و فرض شماره ردیف p برای این بلوک داریم:

اگر $p >= r$ آنگاه اصلاح R قطعا روی **DB** فیزیکی ثبت شده است و

اگر $p < r$ آنگاه اصلاح R قطعا روی **DB** فیزیکی ثبت نشده است

الگوریتم جدید: ص ۱۹,حذف مرحله ۳ و افزودن شماره m به رکورد **chkpnt** مرحله ۲

m شماره ردیف اولین رکورد در بافر که هنوز روی **DB** نوشته نشده (کوچکترین) یا دورترین نقطه ای که باید از آنجا **redo** نمود, امکان وجود نوع T1 جاری, الگوریتم

فصل اول: ترمیم, ۱-۵ شکست محیط ثبت داده ها

مفهوم: معیوب شدن بخشی از حافظه فرعی (دیسک) حاوی **DB**

راه حل: آوردن آخرین کپی DB (تنها راه), تکرار اصلاحات از روی **log** جاری و آرشیو, مشکلات ایجاد کپی یا دامپ **DB**, زمان مشخص و عدم وجود تراکنش جاری, ایجاد دامپ افزایشی در بعضی سیستمها

ترتیب عملیات: مراحل به شرح زیر

۱- وقوع شکست در محیط ثبت داده ها
۲- شکست کلیه تراکنشهای جاری (توقف غیر عادی)
۳- مطلع کردن اپراتور برای جایگزین کردن یک محیط جدید
۴- بازسازی **DB** از روی آخرین **bkup**
۵- تکرار اصلاحات (**redo**) بر مبنای **log** جاری و **log**های آرشیو
لزوم وجود شماره ترتیب در فایلهای **log** بری پیشگیری از خطا
۶- امکان معیوب شدن **log** و لزوم ایجاد دو نسخه از آن در طول عملیات
۷- لزوم **cold start** در صورت از دست دادن کامل **log**

فصل اول: ترمیم, ۱-۶ CMT دو فازی

حالت عادی: خالی کردن بافرهای log قبل از انجام CMT

مسئله: وجود بیش از یک منبع مثلا DB و پیغامها و
لزوم عملکرد مستقل هر منبع (سیاست AoN)

ترتیب عملیات: مراحل CMT دو فازی به شرح زیر

۱- اعلام CMT از طرف یک عضو به هماهنگ کننده (عضو جدید DBMS)
۲- اعلام لزوم آمادگی برای CMT به همه منابع از طرف هماهنگ کننده
۳- اخذ پاسخها از همه منابع و and کردن آنها
۴- اعلام CMT به همه آنها در صورت OK بودن نتیجه
۵- اعلام RB به همه در غیر این صورت
۶- مسئولیت پیگیری امور log در هر منبع با خود منبع

پیاده سازی: شکلهای ۱-۵ و ۱-۶ ص ۲۳

تکلیف فصل اول: مطالعه محیطهای بخش ۱-۷ و دو محیط دیگر, ارائه خلاصه در کلاس
تحویل گزارش کامل