

**Архивиране и възстановяване с MySQL Community Edition:
очаквания и реалност**

Иван Куюмджиев

Backup and Recovery with MySQL Community Edition: Expectations and Reality

Ivan Kuyumdzhiiev

Abstract

This publication explores backup and recovery capabilities of the most popular database management system - MySQL Community Edition. Available backup types are categorized and compared theoretically. Experiments are conducted to measure the capabilities of each backup and recovery method. Knowing these characteristics could help choosing backup type, storage engine (MyISAM or InnoDB) and MySQL settings that would be most suitable for business needs. Conducted experiments combined with analytical approach can lead to building effective backup and recovery strategies.

Keywords: Backup, Recovery, Database, MySQL

JEL Code: C88

Въведение

MySQL е една от най-популярните бази от данни, използвани в Интернет и обслужва хиляди сайтове с различно предназначение. Използва се за както в нови проекти с предимно авторски код (Петров, 2008) и в повечето платформи с отворен код за електронна търговия (Сълова, 2017), така и в популярни работни рамки като системи за блогове (WordPress), системи за управление на съдържанието (Joomla, Drupal), и форуми (phpBB). Списъкът може да бъде разширен (MySQL, 2018d) с клиенти като Facebook, Twitter, Symantec, Verizon, NASA, YouTube, Wikipedia и много други. Според някои автори (Банков, 2018) широкото разпространение на MySQL е породено от факта, че през годините тя е единствената СУБД със своя Community Edition (CE) версия, която е без лицензионна такса за експлоатация според GPL лиценз (GNU General Public License).

Разнообразието от клиенти и широкото приложението на MySQL предполагат, че са налични достатъчно надеждни и ефективни възможности за архивиране и възстановяване на базите от данни, които той управлява. Публикацията има за цел да очертае възможните подходи за извършване на тези дейности по най-оптималния начин от гледна точка на изискванията за минимизиране на загубата на данни и непрекъсваемост на работата на информационната система.

1. Особености на MySQL - видове хранилища

Преди да бъдат разгледани възможностите на MySQL за архивиране и възстановяване е необходимо да се обърне внимание на значим фактор за работата на базата от данни – използваното хранилище. MySQL поддържа архитектура, която позволява избор на различни хранилища на данни за всяка таблица. SQL Engine изпраща заявки към хранилища и те от своя страна изпълняват действия върху съхраняваните в тях данни на физическо ниво, което означава че би следвало да имат влияние върху процесите на архивиране и възстановяване.

Bell (2013) определя използването на различни хранилища в MySQL като една от най-добрите възможности предлагани от продукта, защото това позволява да бъде настроена базата данни като се избира най-подходящото хранилище в зависимост от изискванията на приложенията към всяка таблица. Например използване на хранилище, което осигурява контрол на транзакциите за много активни бази от данни или използване на хранилище, което съхранява данните в оперативната памет в случаи, в които те се достъпват често, но не

се променят.

За да се открие връзката между процесите на архивиране и възстановяване, и използваните хранилищата от данни, ще бъдат изведени характеристиките на най-често използваните от тях.

MySQL 8.0 поддържа следните хранилища за данни: MyISAM, MERGE, MEMORY, InnoDB, CSV (текстови файлове с разделител), Blackhole, Archive, Example (шаблон за създаване на нови хранилища) (MySQL, 2018b)), като представя всяка таблица на твърдия диск като поне един файл. Той има разширение .frm, съдържа описание на структурата на таблицата и се създава от сървър, а различните хранилища могат да създадат допълнителни файлове, съдържащи данните от таблиците и информация за индексите, като имената и структурата им зависи от хранилището (DuBois, 2009, p. 590).

MyISAM е типът хранилище на данни по подразбиране в MySQL до версия 5.5. Таблиците са създадени чрез методи за компресия и оптимизация на индексирани данни и приложения за електронна търговия (Bell, 2013). Заключение е на ниво таблица и изборът на това хранилище се препоръчва, когато скоростта на четене е от най-голямо значение. MyISAM представя всяка таблица като три файла на твърдия диск – освен файла, описващ структурата на таблицата, се използват и файлове с разширение .myd (за запис на съдържанието на редовете) и .myi (запис на информация за индексите) (MySQL, 2018h).

InnoDB поддържа механизъм за превъртане назад на изпълнени до край транзакции, като таблиците от този вид могат да нарастват до 64 TB (по-малко от MyISAM – 256TB). Въпреки по-малкия максимален размер хранилището се използва по подразбиране за всички новосъздадени таблици и от фирмата разработчик (Oracle) препоръчват прилагането му. InnoDB е първото хранилище в MySQL, което поддържа работа с транзакции (съгласуване с принципа ACID) и поддържане на референтна цялост на данните. Таблицы, съхранявани в такъв формат, предлагат заключване на ниво запис (единствено платеното хранилище NDB предлага подобна възможност) – докато се изпълнява транзакция върху дадени записи от таблицата, те остават заключени, но други транзакции могат да се изпълняват върху останалите записи. Считаме, че в това се корени причината според някои изследвания (Vaswani, 2009, p. 53) InnoDB таблиците да съвпадат с (и понякога надминават) производителността на MyISAM таблиците.

От гледна точка създаването на логически архиви е от голямо значение да се оцени производителността на това хранилище в сравнение с другите възможни. В официалната документация (MySQL, 2017a) на MySQL като предимство на InnoDB е посочено, че поддържа собствен буфер, който кешира често достъпваните данни и индекси в оперативната памет, което води до ускоряване на обработката. Според същата документация хранилището е проектирано за процесорна ефективност при обработка на големи обеми от данни (при специализираните сървъри за бази от данни до 80% от оперативната памет е заделена за кеша на InnoDB).

Анализирайки характеристиките на двата най-използвани вида хранилища, считаме че е необходимо да бъде проверена хипотезата, че ако скоростта на четене е по-бърза при MyISAM, а скоростта на запис е по-бърза при InnoDB, това ще се отрази на скоростта на архивиране и възстановяване по следния начин: по-бързо създаване на архиви на таблици от тип MyISAM и по-бързо възстановяване на таблици от тип InnoDB. Ако изведената зависимост е вярна, то е необходимо да се дефинира как това се отразява на общото време за архивиране и възстановяване - дали предимството при запис оказва по-голямо влияние от предимството при четене.

2. Видове архивиране в MySQL Community Edition

За основа при определяне на възможностите на MySQL ще бъде използвана вече направена класификация в друго изследване (Куюмджиев, 2018), според която видовете

архивиране се разделят в зависимост от това каква част от базата данни съдържат, дали могат да се изпълняват по време на работа и дали съхраняват реални данни или команди на 12 вида (таблица 1).

Таблица 1. Видове архиви – оценка на скоростта.

№	Вид архив	Скорост
1	Офлайн физически пълен архив	Много добра
2	Офлайн физически архив с допълване	>1 ¹
3	Офлайн физически архив на разликата	>1
4	Офлайн логически пълен архив	Добра
5	Офлайн логически архив с допълване	<2
6	Офлайн логически архив на разликата	<3
7	Онлайн физически пълен архив	Средна
8	Онлайн физически архив с допълване	>7
9	Онлайн физически архив на разликата	>7
10	Онлайн логически пълен архив	Най-ниска
11	Онлайн логически архив с допълване	>10
12	Онлайн логически архив на разликата	>10

Schwartz (2012, p. 637) свежда възможностите на безплатната версия на MySQL до два вида логически архиви: файлове, съдържащи команди за създаване и добавяне на записи в таблиците (наричани за краткост дъмпове), и текстови файлове с разделител, съдържащи записите от таблиците (екстракцията на таблици).

След анализ на характеристиките им можем да твърдим, че дъмповете представляват онлайн логически пълни архиви на база от данни в MySQL създавани с помощта на инструмента mysqldump. Mysqldmp генерира SQL код за създаване на структурата на таблиците, последван от код за добавяне на данни в тях чрез INSERT оператори. По подразбиране mysqldump заключва всички таблици на базата от данни преди да бъдат архивирани, за да се предотврати несъстоятелност на архива при поява на конкурентни модификации в базата по време на операцията (DuBois, 2014, p. 149).

При възстановяване също се заключват таблиците, за да не могат да бъдат променяни по време на изпълнение и като допълнение се използва ALTER TABLE, за да се изключат временно индексите за таблиците. След приключване на добавянето се включва индексирането и се отключват таблиците (Murach, 2012, p. 532). Избраната стратегия за заключване е от изключително значение за целия процес на архивиране и възстановяване, и определя дали приложенията ще могат да извършват запис в базата от данни по време на създаването на архива.

Възможни са няколко варианта за ограничаване на достъпа до таблиците при създаване на архиви с Mysqldump.

Заклучването с --lock-tables не гарантира възстановяване в логически вярно състояние. Опцията заключва всички таблици от една схема, а не всички таблици във всички схеми. Ако приложението записва в две различни схеми и хранилището не поддържа транзакции (например MyISAM) е възможно да се получи загуба на цялост на данните по време на архивиране.

Опцията --lock-all-tables изпълнява командата FLUSH TABLES WITH READ LOCK, за да осигури цялостност на данните във всички схеми. Тази команда инструктира сървъра да

¹ Да се чете – скорост по-добра от скоростта на архив с номер 1 (офлайн физически пълен архив)

изчаква да приключат всички транзакции и затваря всички отворени таблици (MySQL, 2018e). Според нас използването на тази опция превръща Mysqldump от онлайн в офлайн логически архив, тъй като базата от данни е недостъпна до приключване на операцията. По тази причина считаме, че би следвало да бъде изпълняван единствено в непроизводствена среда при невъзможност да се изпълни офлайн физическо архивиране, което би отнело значително по-малко време и системни ресурси.

При използване на MySQL невинаги е нужно заключване на таблиците за постигане на цялост на данните. Опцията --single-transaction стартира транзакция и гарантира, че таблиците от тип InnoDB се архивират със запазена цялост на данните (Bradford, 2012, p.23), но това не важи за таблици от вид MyISAM и Memory (MySQL, 2018f). Командата е валидна само за хранилища като InnoDB, при които се съхранява информация за стойности на променяните редове. Тъй като архивът се превръща в една голяма транзакция, този подход добавя натоварване на системата и би следвало да увеличи времето за създаване и възстановяване от архива, но да гарантира логическата цялост на данните. За подобряване на производителността при архивиране на големи таблици с опцията --single-transaction Bradford (2012, p. 24) препоръчва добавяне на опцията --quick, която инструктира MySQL да извлича записите един по един директно в архива, вместо първо да се извлече цялата таблица в оперативната памет, преди да бъде записана ред по ред в архива.

Налага се изводът, че колкото е по-голяма базата от данни, толкова по-голям ще става резултатният файл и прогресивно ще се увеличава времето, нужно за създаване и възстановяване на архива. Въпреки че инструментът няма ограничение за размера на базата от данни, според горепосоченото изследване, използването му в зависимост от наличният хардуер, е препоръчително за размери между 5 и 20 GB. Възстановяването с mysqldump предполага изпълнение на SQL командите в архива, което включва входно-изходни операции при добавяне, създаване на индекси и т.н., което, както беше посочено по-горе, води до ниски резултати. За по-големи бази от данни официалната документация (MySQL, 2018g) препоръчва използване на физически архиви на файловете или инструмента mysqlbackup от MySQL Enterprise Backup.

Екстракцията на таблици (в текстови файлове с табулация за разделител на колоните и символ за нов ред за разделител на редовете) е метод, който предоставя по-бързо архивиране и възстановяване от архивите, създадени с mysqldump (табл 2.3.2) (Bradford, 2012, p. 39).

Таблица 2. Сравнение на резултати на mysqldump и екстракция на таблици.

Метод	Размер на дъмпа	Време за архивиране	Време за възстановяване
mysqldump	727 MB	102 сек.	600 сек.
Екстракция	669 MB	86 сек.	301 сек.

Този подход създава текстов ASCII файл със съдържанието на всяка таблица от базата данни, без коментари и SQL команди. При така описаните характеристики методът може да бъде причислен по-скоро към физическите, отколкото към логическите архиви, защото съдържа данните от таблиците, а не команди, които да забавят създаването и възстановяването.

Екстракцията на таблици се свежда до изпълнение на заявки до всяка една от таблиците и записването на резултата от тях в посочен файл. Не са предвидени заключвания на таблици по време на изпълнението на командите, което означава, че докато се създаде архив за поредната таблица, базата от данни може да бъде променена многократно, което ще доведе до грешно съдържание на архива. По тази причина считаме, че екстракцията е метод, който е подходящ само за статични таблици (съхраняващи историческа информация).

Въпреки, че предоставя по-добро време за архивиране и възстановяване от `mysqldump`, използването му само със статични таблици изключва възможността да бъде приложен за създаване на пълен архив на базата от данни. Тъй като не предоставя и методи за възстановяване към точен момент от време, считаме, че може да бъде използван единствено в хибридна стратегия за архивиране. За тази цел следва първо да бъдат определени таблици със статично съдържание, които не влияят на цялостта на данните в другите таблици, за да бъдат архивирани чрез него.

Като други недостатъци на този подход някои автори (Schwartz, Zaitsev, Tkachenko, 2012, p. 638) посочват:

- Архивът може да се създаде само във файл, намиращ се на машината на която работи инстанцията на MySQL;
- MySQL трябва да има позволения за запис в директорията, където се записва файлът, защото MySQL създава файла, а не потребителят, който изпълнява командата;
- Файлът не може да бъде презаписван от MySQL – за изпълнение на командата е задължително да не съществува файл с такова име;
- Файлът не може да бъде създаден директно в компресиран вид (възможно е да се добави компресия след създаване);
- При наличие на нестандартни символи е възможно възникване на грешки при архивиране или възстановяване.

По тези причини посоченият метод няма да бъде разглеждан като самостоятелен елемент от стратегия за архивиране и възстановяване.

Други автори (Murach, 2012) добавят и възможност за извършване на архиви с допълване в MySQL. Архивите с допълване съдържат всичката информация за промените по базата от данни, възникнали от последния архив до момента. Файловете от бинарния лог осигуряват информацията нужна за пресъздаване на действията, възникнали с базата данни от момента на последния архив.

Създаването на подобен вид архиви се изразява в последователност от няколко действия. При създаване на нов архив с допълване трябва да се даде команда за записване на всички последващи операции в нов файл (FLUSH LOGS) (MySQL, 2018c). По този начин всички предхождащи команди ще се съдържат в наличните файлове на бинарния лог, а всеки последващи – в новосъздадените. При следващия пълен архив с `mysqldump` също трябва да се изпълни командата за записване на всички последващи операции в нов файл (`mysqldump -flush-logs`).

Наличието на файлове, съдържащи операциите извършвани по базата от данни в същата директория, в която се намират и използваните файлове, не може да бъде окачествено като надежден архив. За да се съхранят архивите на бинарния лог е възможно да се използват команди от операционната система за директно копиране, инструменти за синхронизиране на съдържанието на папки или инструментът `mysqlbinlog`. Отново според официалната документация (MySQL, 2018i) във версия MySQL 8, `mysqlbinlog` може да чете лога на операциите и освен да показва неговото съдържание в текстов вид, да го записва в двоичен вид в друг файл. Тази възможност позволява лесно архивиране на лога в оригиналния му формат.

Съобразно разгледаните характеристики може да бъде направен изводът, че безплатната версия на MySQL предоставя следните видове онлайн архивиране на бази от данни:

- онлайн логически пълен архив чрез `mysqldump`;
- онлайн логически архив с допълване чрез архивиране на бинарния лог.

Извършването на офлайн архиви или архиви от тип момента снимка са извън изследваната предметна област, а екстракцията на статични таблици не може да бъде причислена към методите за онлайн архивиране, поради предимно динамичната структура на

базите от данни.

Налага се изводът, че безплатната версия на MySQL предлага силно ограничени възможности за архивиране и не предоставя инструменти за създаване на:

- онлайн физически пълен архив,
- онлайн физически архив с допълване,
- онлайн физически архив на разликата,
- онлайн логически архив на разликата,

което ограничава възможните стратегии за архивиране и възстановяване чрез инструментите на MySQL до два варианта:

1) Стратегия пълни архиви – изпълнение на `mysqldump` на определен времеви интервал.

2) Стратегия пълни архиви и архиви с допълване:

1. Създаване на пълен архив на базата с `mysqldump`.
2. При включен бинарен лог всички операции от този момент се записват в него.
3. На определен интервал текущият лог се затваря и се копира на сигурно място.

Създава се нов лог, който да записва следващите операции.

4. Стъпка 3 се повтаря според нуждите на базата от данни.

Настройката на бинарния лог (`--binlog-format`) в MySQL определя как да бъдат записвани данните в него – като SQL заявки, стойности на променяни редове или комбинация от двете.

Ако при архивиране и възстановяване се копират стойностите на променяните редове, а не се преобразуват към SQL команди, които да бъдат изпълнени, се налага изводът, че вторият метод за управление на бинарния лог би следвало да доведе до по-добри показатели. За да бъде проверена тази хипотеза считаме за целесъобразно да се изпълнят следните тестове:

- Стъпка 1. Задаване на опция `--binlog-format` стойност;
- Стъпка 2. Увеличаване размера на базата от данни;
- Стъпка 3. Създаване на архив;
- Стъпка 4. Възстановяване на архива от точка 3;
- Стъпка 5. Връщане в точка 2.

Този процес се повтаря за трите възможни настройки, като се използва едно и също увеличаване размера на базата от данни.

Към тези тестове считаме за необходимо да се добавят и такива, измерващи характеристиките на архивиране и възстановяване при използване на различно хранилище на данни. За да се избегне излишно превключване от едното хранилище към другото, предлагаме първо да бъдат изпълнени стъпките за MyISAM, след това да се възстанови базата от данни към начално положение, да се конвертира и да се повторят тестовете за InnoDB. По този начин последователността от стъпки ще добие следния вид:

- Стъпка 1. Увеличаване размера на базата от данни;
 - Стъпка 2. Архивиране на базата от данни с таблици от тип MyISAM;
 - Стъпка 3. Възстановяване от архива от точка 2;
 - Стъпка 4. Обратно в стъпка 1.
- Същият тест да се повтори и за базата от данни с таблици InnoDB.

3. Зависимости между вида на хранилището в MySQL, настройката на лога и ефективността на архивирането и възстановяването

Резултатите от двата теста са обединени, защото имат обща последователност от стъпки. Като цел на експериментите е да се открие зависимост между вида на хранилището в MySQL, настройката на лога и ефективността на архивирането и възстановяването.

За тази цел са изпълнени четири експеримента, като резултатите ще бъдат групирани

в две групи (съответно при архивиране и възстановяване (таблица 2)) по четири категории:

1. Хранилище InnoDB binlog-format statement
2. Хранилище MyISAM
3. Хранилище InnoDB binlog-format row
4. Хранилище InnoDB binlog-format mixed

Таблица 2. Сравнение на средно време (в сек.) за изпълнение на архиви в MySQL.

	InnoDB, statement	MyISAM	InnoDB, row	InnoDB, mixed
Архивиране	5,33	4,06	5,01	5,03
Възстановяване	99,49	19,22	92,76	95,35

Не представлява изненада фактът, че използването на настройка на бинарния лог, при която той съдържа SQL заявките към сървъра е с най-ниски показатели както при архивиране, така и при възстановяване на базата от данни. Анализът на резултатите от сравнение на ефективността на архивирането при различни настройки на бинарния лог сочи, че най-добри показатели се получават, когато той съдържа променените редове в таблиците.

Въпреки, че скоростта на запис на данните при използване на хранилище InnoDB е по-бърза от тази при MyISAM, се наблюдава значителна разлика в резултатите от архивиране при двете хранилища. Според хипотезата в първа точка, скоростта на създаване на архив при хранилище MyISAM би следвало да е по-добра, заради по-високата скорост на четене на данните. В проведените тестове времето необходимо за създаване на архив при използване на MyISAM е с 20% по-малко от това при InnoDB, но също така времето за възстановяване на този архив пет пъти по-добро от това при използване на по-новото от двете хранилища.

В резултат на проведените експерименти могат да бъдат направени следните изводи:

1) Ефективността на инструмента mysqldump е пет пъти по-добра (както при архивиране, така и при възстановяване) при използване на хранилището MyISAM вместо InnoDB.

2) При използване на хранилище InnoDB най-ефективно архивиране и възстановяване се постига при използване на настройка на лога, при която той съдържа променените редове в таблиците.

По теоретичен път беше стигнато до изводите, че екстракцията на таблици е метод подходящ само за статични таблици и по този начин единствените методи за онлайн архивиране в безплатната версия на MySQL са логически пълен архив чрез mysqldump и логически архив с допълване чрез архивиране на бинарния лог.

Заклучение

Процедурите по архивиране и възстановяване на бази от данни заемат важно място в административните задачи свързани с поддържане информационните системи в ефективно и работещо състояние. Времето необходимо за създаване на архивния набор оказва влияние на натоварването на системата, но скоростта на възстановяване и количеството загубена информация са с още по-голямо значение, защото могат да дефинират границата между съществуващ и несъществуващ бизнес. По тези причини, всеки администратор на бази от данни трябва да може да избере подходящ план за архивиране и възстановяване. Настоящото изследване очертава силни и слаби страни на вградените възможности на MySQL Server Community Edition – една от най-разпространените системи за управление на бази от данни. Направените изводи могат да бъдат от полза при изграждане на стратегия за архивиране и възстановяване на нови бази от данни или актуализиране на вече съществуващи с цел намаляване времето за извършване на тези процеси и намаляване на евентуалните загуби на информация.

Използвана литература

1. Банков, Б. (2018) Софтуерна система за автоматизирана обработка на неструктурирани данни от социалните мрежи. ИУ-Варна
2. Куюмджиев, И. (2018) Методологически и технологични аспекти при архивирането на бази от данни. Цани Калянджиев
3. Петров, П. (2008) Сървърно програмиране. Варна: Наука и икономика.
4. Сълова, С. (2017) Тенденции в развитието на софтуерните платформи с отворен код за създаване на електронни магазини. Известия на Съюза на учените - Варна, Серия Икономически науки, Варна: Съюз на учените, 2, 187 - 196
5. Bell, C. (2013) Expert MySQL. Apress
6. Bradford, R. (2012) Effective MySQL Backup and Recovery. McGraw Hill Professional.
7. DuBois, P. (2009) MySQL. Fourth Edition. Addison-Wesley
8. DuBois, P. (2014) MySQL Cookbook: Solutions for Database Developers and Administrators. O'Reilly Media, Inc.
9. Murach, J. (2012) Murach's MySQL. Mike Murach & Associates, Incorporated
10. MySQL (2017a) Benefits of Using InnoDB Tables. [Online] Available from: <http://dev.mysql.com/doc/refman/5.7/en/innodb-benefits.html> [Accessed: 16/10/2018]
11. MySQL (2018b) Chapter 16 Alternative Storage Engines [Online] Available from: <https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html> [Accessed: 16/10/2018]
12. MySQL (2018c) Database Backup Methods. [Online] Available from: <http://dev.mysql.com/doc/refman/8.0/en/backup-methods.html> [Accessed: 16/10/2018]
13. MySQL (2018d) MySQL Customers. [Online] Available from: <https://www.mysql.com/customers/> [Accessed: 16/12/2017]
14. MySQL (2018e) FLUSH Syntax. [Online] Available from: <https://dev.mysql.com/doc/refman/8.0/en/flush.html> [Accessed: 16/10/2018]
15. MySQL (2018f) mysqldump — A Database Backup Program. [Online] Available from: http://dev.mysql.com/doc/refman/8.0/en/mysqldump.html#option_mysqldump_single-transaction [Accessed: 16/12/2017]
16. MySQL (2018g) mysqldump — Performance and Scalability Considerations. [Online] Available from: <https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html#mysqldump-performance> [Accessed: 11/07/2018]
17. MySQL (2018h) The MyISAM Storage Engine. [Online] Available from: <https://dev.mysql.com/doc/refman/8.0/en/myisam-storage-engine.html> [Accessed: 16/10/2018]
18. MySQL (2018i) Using mysqlbinlog to Back Up Binary Log Files. [Online] Available from: <https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog-backup.html> [Accessed: 16/10/2018]
- mysqldump-secure (2017) Live builds. [Online] Available from: <https://mysqldump-secure.org/> [Accessed: 16/10/2018]
19. Schwartz, B., Zaitsev, P., Tkachenko, V. (2012) High Performance MySQL: Optimization, Backups, and Replication. O'Reilly Media, Inc
20. Vaswani, V. (2009) MySQL Database Usage & Administration. McGraw Hill Professional

За контакти

гл. ас. д-р Иван Куюмджиев
Икономически университет - Варна
ivan_ognyanov@ue-varna.bg.com