International Journal of Innovative Engineering Applications

Journal homepage: https://dergipark.org.tr/ijiea

DATABASE RECOVERY TECHNIQUES IN MICROSOFT SQL SERVER

***Ender Şahinaslan*[1], *Önder Şahinaslan*[*2]**

[1]*EA Health, Education and Informatics Ltd. Sti., IT Consultant, Turkey*
[2]*Maltepe University, Head of Informatics Department,, Turkey*

Abstract

Original scientific paper

In today's world where information-based big data is shared, protecting, storing and accessing data is of critical importance. Protected data should be accessible when needed. Digitization and digital transformation cause great changes in data storage and storage technologies. While the size of data backup tools is getting smaller, their capacity is increasing, and they are moving towards more cost-effective and faster reliable technologies. Despite all these positive developments, there are problems of corruption on the devices and systems where the data is stored. As a result, data access problems occur. This problem means that business processes stop in an institution where data-based transactions take place. The damage to the drying of each time spent in the study is quite large. On the other hand, in sectors such as banking, education, defense, health, insurance, agriculture, telecommunications, data is usually kept on one or more of the relational databases such as Microsoft SQL, My SQL, Postgre SQL and Oracle. There is a need for new research on how to recover data with which techniques, processes and methods in the face of a corruption event that may occur on these databases or servers. This issue has a critical importance in terms of ensuring the continuity of digital data infrastructures. If a quick solution cannot be produced in the face of a sudden interruption in database access, it will cause serious problems in business continuity. The causes of corruption in the use of Microsoft database, the measures that can be taken against them, consistency checks and database recovery methods are investigated. Techniques and methods based on database recovery scenarios in Microsoft SQL server used in large data centers are examined. Methods of overcoming a corruption problem in the database with the least damage in the shortest time are explained, and application practices are studied through VT recovery scenarios. This study also serves as a guide for students, researchers and technical staff.

Keywords: *MS SQL server, corruption, database, database recovery, data backup.*

MICROSOFT SQL SUNUCUSUNDA VERİ TABANI KURTARMA TEKNİKLERİ

Özet

Orijinal bilimsel makale

Enformasyona dayalı büyük verilerin paylaşıldığı günümüzde verinin korunması, saklanması ve erişilmesi kritik öneme sahiptir. Korunan bir veri ihtiyaç duyulduğunda erişilebilir olmalıdır. Dijitalleşme ve dijital dönüşüm veri saklama ve depolama teknolojilerinde büyük değişimlere neden olmaktadır. Veri yedek alma araçlarının boyutları küçülürken kapasiteleri yükselmekte, daha uygun maliyette ve daha hızlı güvenilir teknolojilere doğru ilerlemektedir. Tüm bu olumlu gelişmelere rağmen verilerin saklandığı cihaz ve sistemler üzerinde bozulma problemleri yaşanmaktadır. Bunun sonucunda veri erişim sorunları oluşmaktadır. Bu sorun veriye dayalı işlemlerin gerçekleştiği bir kurumda iş süreçlerinin durması demektir. Çalışmada geçen her bir sürenin kuruma vereceği hasar oldukça büyüktür. Diğer taraftan bankacılık, eğitim, savunma, sağlık, sigorta, tarım, telekomünikasyon gibi sektörlerde veriler genellikle Microsoft SQL, My SQL, Postgre SQL ve Oracle benzeri ilişkisel veri tabanlarının biri veya birkaçı üzerinde tutulur. Bu veri tabanları veya sunucular üzerinde yaşanabilecek bir bozulma olayı karşısında verinin hangi teknik, süreç ve yöntemlerle nasıl kurtarılması konusunda yeni araştırmalara ihtiyaç vardır. Bu konu dijital veri altyapıların sürekliliğinin sağlanması bakımından kritik öneme sahiptir. Veri tabanı erişiminde yaşanabilecek ani bir kesinti karşısında hızlı bir çözüm üretilememesi durumunda iş sürekliliğinde ciddi sorunlara neden olur. Microsoft veri tabanı kullanımında karşılaşılan bozulma nedenleri, bunlara karşı alınabilecek önlemler, tutarlılık kontrolleri ve veri tabanı kurtarma yöntemleri araştırılmıştır. Büyük veri merkezlerinde kullanılan Microsoft SQL sunucusunda veri tabanı kurtarma senaryolarına dayalı teknik ve yöntemler incelenmiştir. Veri tabanında yaşanabilecek bir bozulma problemin en kısa sürede en az hasarla atlatabilmenin yöntemleri anlatılmış, VT kurtarma senaryoları üzerinden uygulama pratikleri çalışılmıştır. Bu çalışma aynı zamanda öğrenci, araştırmacı ve teknik çalışanlar için bir kılavuz niteliğindedir.

Anahtar Kelimeler: *MS SQL sunucu, bozulma, veri tabanı, veri tabanı kurtarma, veri yedekleme.*

* Corresponding author.
E-mail address: ondersahinaslan@maltepe.edu.tr (Ö. Şahinaslan)
Received 10 February 2022; Received in revised form 06 June 2022; Accepted 20 June 2022
 Doi: https://doi.org/10.46460/ijiea.1070325

1 Giriş

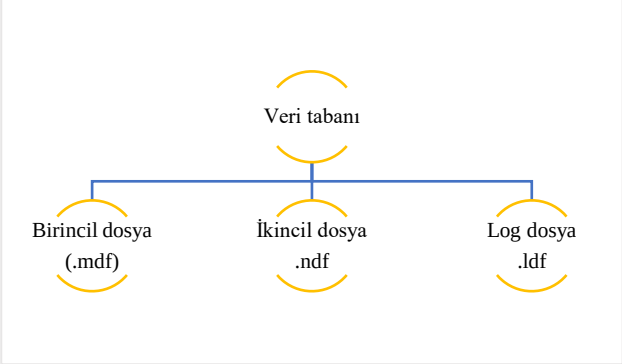
Veri günümüzde en önemli varlıklardan biri haline gelmiştir. Veri, harf, rakam, sembol ve işaretlerden oluşan ham, işlenmemiş gözlem veya gerçeklerdir [1]. Veriler video, kâğıt, dosya, disk, veri tabanı(VT) gibi ortamlarda farklı biçimlerde yer alır. Dijitalleşmeye bağlı olarak veriye olan talep, ihtiyaç ve kullanım biçimleri bilgi varlıklarının merkezinde konumlanmıştır. Veriden bilgiye dönüşüm süreçlerinde ve hızlı stratejik kararlar almada güncel veriler kullanılır. Bu verilere farklı platformlardan aynı anda erişme, okuma, işleme ve yazma işlemleri gerçekleştirilir. Birbirleriyle veri alış verişinin yaygın hale geldiği günümüzde veriler genellikle merkezi yapılar üzerinde güvenli ve erişilebilir halde tutulmaya çalışılır. Bu ihtiyacı karşılayan en önemli platformlar veri tabanlarıdır. VT, kendi aralarında mantıksal olarak ilişkilendirilmiş verilerin belirli bir sistematik yapı içerisinde çok amaçlı kullanımına olanak sağlayan yapılardır [2]. Oluşturulan merkezi veri tabanları sayesinde her türlü uygulama, donanım, sistem araçları üzerinden elde edilen veriler kaydedilir, işlenir talep edildiğinde hızlı ve güvenli bir biçimde iletilir. Anlık tutulan kayıtlar zamanla artar ve büyük kapasiteli veri tabanları haline dönüşür. Geçmişe dönük kaydedilen bu verilerin analizinden şirketler küresel rekabet ortamında hizmet seviyelerinin yükseltilmesinde, kalite ve maliyetlerin iyileştirilmesinde birçok alanda avantajlar sağlamaktadır. Dolasıyla işletmelerde tutulan verilerin kapasiteleri sürekli artmaktadır.

İş ve yaşam biçimlerinin hızla dijitalleşmesi, Endüstri 4.0'a geçiş ve bilgi teknolojilerindeki artış bilgiye olan ihtiyacı artırmaktadır. Diğer taraftan telekomünikasyon, bankacılık, sağlık, savunma, eğitim gibi birçok sektörde iş sürekliliğini etkileyebilecek anlık kesintiler yâda veri kayıpları sistemlerin sürekliliği üzerinde yıkıcı oluşturur. Bir yandan veriye be denli bağımlılık söz konusu iken diğer taraftan veri tabanlarına gün içinde yaşan teknik problemlerden dolayı erişilememe ve yedekten dönme ihtiyacı bir gerçekliktir. Teknik olarak bu bozulma durumları tam olarak önlenemese de bir veri bozulması söz konusu olduğunda bunun en az kayıpla en hızlı bir şekilde yeniden işler hale getirilebilmesinin yol ve yöntemlerinin neler olduğu bir araştırma ve çalışmanın konusudur. Verilen örnek senaryo ve anlatımlarda uygulanacak yol ve yöntemlerin ne biçimde olması gerektiği, öncesinde yapılacak testlerle belli bir deneyimin elde edilmesi, olası kesinti veya bozulma karşısında ise aksiyonların en hızlı alınarak sürecin doğru şekilde yönetilmesine katkı sağlayacaktır.

Veriler veya verilerin depolandığı donanım, sistem ve uygulamalarda zamanla çeşitli bozulmalar yaşanmaktadır. Bu bozulmalar sonucunda normal yöntemlerle erişilemeyen verilerin özel yöntemlerle geri getirilmesine işlemine veri kurtarma denir [3]. Bu verilerin kurtarılması konusunda, ticari firmaların kullandıkları araç ve yöntemler gizlilik ve ticari rekabet gibi çeşitli nedenlerden dolayı paylaşılmamaktadır [4]. Bu durum, her hangi bir sebepten dolayı VT üzerinde yaşanacak bir veri bozulma olayının gerçekleşmesi durumunda bu tür ticari firmaların kapalı uygulamalarına bağımlı bırakmaktadır. Oysa çözüme dair temel yöntem ve tekniklerin, bunlara ilişkin işlem adımlarının belirli olgunluk ve düzende bir araya getirilerek uygulanabilir bir yapının açık biçimde ortaya çıkartılma ihtiyacı vardır. Diğer taraftan dünyanın önde gelen veri kurtarma hizmeti sağlayıcılarından birisi olan Ontrack firması tarafından Almanya, Amerika Birleşik Devletleri, Fransa, Kanada, İngiltere, İspanya ve İtalya'dan 484 kuruluşla anket yapılmıştır. Bu anket sonuçlarına göre ankete katılan kuruluşların %39'unun herhangi bir fidye yazılımı acil durum plana sahip olmadıklarını veya bilmediklerini ifade etmişlerdir. Ankete katılanların %21'i bir fidye yazılımı saldırısı yaşadıklarını ve bu olay sonrası çalışabilen bir yedekleme ünitesine erişemediklerini söyleyenlerin oranı ise %26'dan fazla bulunmuştur. Ayrıca çalışan bir yedeklemeye erişebilenlerin ise yalnızca %22'si verilerin sadece bir kısmi geri yükleyebildiğini veya hiç geri yükleyemediği yönünde görüş vermişlerdir [5]. Bu sonuçlar saldırıların verilere erişememe problemine neden olduğunu, bu tür sorunların gelişmiş ülkelerin belli başlı kurumlarında bile kayda değer ölçekte yaşanabildiğini göstermektedir. Bu örneklerde veri erişim problemi yanında verinin şifrelenmemiş ilk haline ulaşamama sorunu da mevcuttur. Saldırı sonrası orijinal veriye dönüşteki başarı düzeyinin düşüklüğü dikkat çekicidir. Bu tür öngörülmeyen saldırı, güvenlik, donanım, yazılım ve sistem altyapı arızaları gibi nedenden dolayı verilere her hangi bir t zamanında erişilememesi söz konusudur. Bu türden bir problemli durumlar sonrası bir hasarla karşılaşılmaması için VT sistemlerinde yedekleme ve kurtarma teknikleri uygulanır. Bu tekniklerin etkin bir şekilde uygulanmasında VT yönetim sistemlerinin üzerinde çalıştığı işletim sistemleri, disk üniteleri etkilidir. Son vd. (2020) yedekleme, kurtarma geri dönüş sürecini kısaltmada SATA tabanlı disk yerine yüksek performanslı SSD disk kullanmanın önemi vurgulanmıştır [6]. Armoush vd.(2020), yedeklenecek verilerin sabit uzunlukta bloklara bölünerek hiyerarşik bir yapı oluşturma, dosya aktarımlarında ise sadece değişen blokların kopyalanarak yüksek güvenilirlik ve hızda bir yedekleme çalışması önerilmiştir [7, 8]. Wibowo vd.(2018) ise yedekleme sunucularındaki yükün azaltılmasına yönelik iş sürekliliğinin sağlanması için yedekleme yönetimi ve performans yönetimini bir arada ölçeklenebilir faktörlerle yönetilmesine vurgu yapılmışlardır [9]. Verilerin sistematik şekilde yedeklenmesi ve geri yüklemesi [10], yedekleme ve kurtarma işlemlerinde verilerdeki tutarlılık [11], veri kurtarmada kullanılan gelişmiş yazılımlar [12], karşılaşılabilecek donanım ve yazılım problemleri [13], sanallaştırmanın önemi, sağladığı performans ve hız kazanımları [14] ile bu süreçlerin etkin biçimde yönetilmesine [15] yönelik çalışmalar başarılı yayınlar olarak incelenmiştir. Tüm bu çalışmalar VT ve kurtarma tekniklerinin bilgi sistemlerine erişilebilirlik ve süreklilik bakımından önemli bir ihtiyaç olduğunu göstermektedir. Yapılan bu araştırmada yaygın kullanıma sahip olan Microsoft SQL (MS SQL) VT sunucusu üzerinde yaşanabilecek bir bozulma karşısında veri kurtarma tekniklerinin neler olduğuna dair elde edilen bilgi ve deneyimlerin uygulamalı olarak anlatılmıştır. Bu çalışma kapsamında VT bozulma nedenleri, veri tutarlılık kontrolleri, kurtarma yöntemleri, fiziksel mimari yapı ve ilgili teknoloji örnekleri, RAID sanallaştırma teknolojisi, VT kurtarma senaryo örnekleri, yedekten dönme işlemleri, kurgulanmış bozulma senaryo ve kurtarma teknikleri MS SQL sözdizimi örnekleri birlikte ele alınmıştır.

2 MS SQL Veri Tabanı ve Bozulma

MS SQL sunucusunda veriler kullanıcılar tarafından görülebilecek şekilde mantıksal bileşenler halinde VT üzerinde tutulur. Fiziksel olarak ise disk üzerinde yer alan veri dosyaları üzerinde tutulur. SQL Server VT'nda üç tür dosya bulunur. Birincil veri dosyasının varsayılan uzantısı '.mdf', ikincil veri dosyalarının uzantısı ise '.ndf'dir. VT kurtarmada kullanılacak günlük bilgilerinin tutulduğu '.ldf' uzantılı 'log' dosyası vardır. Şekil 1'de MS SQL VT dosya türleri gösterilmektedir.



Şekil 1. MS SQL VT dosya türleri.

MS SQL VT'ndan veriler SQL Server sunucusu üzerinde '*mdf*' uzantılı veri dosyaların sayfa (*page*) adı verilen küçük birimler üzerinde tutulur. Bir veri üzerinde okuma, değiştirme ve güncelleme yapılmak istenildiğinde VT motoru (*database engine*) öncelikle ilgili verinin bulunduğu sayfayı ana bellek (*memory*) üzerinde arar ve ilgili verinin bellek üzerinde bulunamaması durumunda ilgili sayfayı disk üzerinden okuyarak ana belleğe kopyalar. Ana bellek üzerinde istenilen değişikliği gerçekleştirdikten sonra ilgili veri sayfasını disk üzerine yazar. Her bir verinin okunma, güncellenme, silme ve kaydedilme temel işleyişi buna benzerdir. Bu DML ve DLL işlemleri esnasında disk ve bellek üzerindeki gidip gelen sayfalar birbiriyle karşılaştırmaya tabi tutulur. Bu karşılaştırma işlemleri sırasında sorgu motoru tarafından her hangi uyuşmazlık tespiti olması durumunda bu bozulma(*corruption*) olarak rapor edilir. Bozulma bellek, disk, veri sayfalarının tutulduğu 'mdf' türü VT dosyalarda ve her işleme ait denetim izlerinin tutulduğu 'log' türü dosyalarda oluşabilir.

2.1 Veri Tabanı Bozuma Nedenleri

VT bozulmalarının birçok sebebi vardır. Bunlardan en sık karşılaşılan sebep disklerden kaynaklıdır. Diskler üzerinde aşırı sıcaklık artışı veya düşüşünde yaşanabilecek değişiklikler, manyetik alana maruz kalma, titreşim, güç kesintileri gibi bir takım nedenlerden dolayı bozulabilmektedir. Verinin depolandığı diskte yaşanan bozulma kaçınılmaz olarak VT/veri bozulmasına neden olmaktadır. VT bozulmasına neden olan diğer etmenler aşağıda aktarılmaktadır.

Disk Şifreleme: VT üzerinde yer alan verilere kontrolsüz veya izinsiz erişimleri engellemek için SQL VT üzerinde şifreleme yöntemi uygulanır. Bunun yerine veya buna ilave diskin şifrelenmesi söz konusu olabilir. Ancak milyonlarca verinin aynı anda okunması, yazılması, güncellemesi ve silinmesi gibi işlemlerin aynı anda zamanla yarışırcasına hızlı gerçekleştiği, çok yoğun kullanılan bir VT kullanımında; VT şifreleme yerine ve buna ek bir disk şifreleme işleminin yapılmaması önerilmektedir. Çünkü her bir şifreleme ek bir süre maliyeti ve işlemlerde gecikmelere neden olmaktadır. Çok yoğun kullanılan VT ve disklerde veri şifresi çözme/işlem/yeniden şifreleme türünde fazladan yapılan işlemler sırasında zaman zaman veri kaçırma veya bozulma olayıyla karşılaşılabilmektedir. Özellikle canlı ortamlarda çalışan veri tabanlarında disk şifreleme önerilmemektedir.

Bellek: VT verilerine ait her türlü işlemin bellek üzerinde yapılır. SQL sunucunun açılıp çalışmasına yetecek yeterli bellek alanı olmaması. Bellek kartı ve bellek çevre birimleri üzerinde oluşan sorunlar.

İşletim Sistemi: İşletim sistemine ait sistemsel hatalar, güncel olmayan işletim sistemlerinde oluşan '*bug*' türü sorunlar, donanım sistem yazılımlarındaki uyuşmazlıklar, SQL Server haberleşme problemleri gibi temelde işletim sisteminden kaynaklı sorunlar

Zararlı Yazılımlar: Virüs gibi zararlı yazılımlar tarafından işletim sistemi, disk, bellek gibi ortamlar üzerinde oluşan hasarlardan kaynaklı VT sistemleri etkilenmektedir. Ancak VT dosyaları, dosya temel alan okuma yapıldığından dosyadaki verilerin etkilenmemesi için anti-virüs uygulamaları üzerinde yapılacak bir ayarlamada VT dosyalarının hariç tutulması özellikle hız performans açısından önerilmektedir.

Elektrik Kesintileri: Özelikle çoklu işlemlerin çevrimiçi gerçekleştiği kritik bir VT'nda bir sayfanın diske yazımı sırasında yaşanabilecek bir elektrik kesintisinin oluşması verinin tam olarak VT depolama sistemi üzerine yazılamamasına sebep olur. Bu ani kesintiler işletim sistemi disk ve VT sistemlerinde üzerinde bozulmalara neden olabilmektedir.

Diğer Nedenler: SAN Controller, ağ('*network*') kartları giriş çıkış(I/O) haberleşme üniteleri, Raid kontrol kartları, yetersiz disk, .mdf, .ldf dosya yapılarında ve disk alanında bozulma(*bad sector*) oluşmasından kaynaklı sorun ve hatalar. Ayrıca aşırı ısınma, sabotaj, darbe, düşürme, cihaz üzerine sıvı dökülmesi, tozlu ortam, manyetik ortam, su baskını, toprak kayması, yangın, deprem, sarsıntı gibi her türlü fiziksel kaynaklı hasarlar ve kasıtlı/bilinçsiz kullanıcıdan kaynaklı kullanım hataları sayılabilir. Tüm bu ve benzeri nedenler MS SQL VT sistemlerini olumsuz etkileyerek veri bozulmalarına neden olabilmektedir.

2.2 Bozulma Öncesi Kontrol ve Önlemler

Her ne kadar VT bozulmalarının önüne tam olarak geçilemeyecek olsa da öncesinde bazı önlem ve kontroller alabilmek mümkündür. Bu kontroller sayesinde olası hata ve bozulmalara karşı tedbir olarak yedekli bir yapıda bir takım uyarı mekanizmaları önceden oluşturulabilir. İzleme ve uyarı araçlarının yardımıyla risklere karşı çeşitli önlemler alabilmek mümkündür. Veri depolama ve yedeklemede MS SQL Sunucu '*fail over cluster*', '*log shipping*', '*replication*', VT aynalama, '*always on*' mimari yapıları ve '*RAID*' sanallaştırma teknolojisi, sunucu hata günlükleri üzerinden giriş/çıkış sistemleri hata kontrolleri ve sunucu kontrol mekanizmalarından sayfa koruma yaygın olarak kullanılan yöntemlerdir.

2.2.1 Giriş Çıkış Hata Kontrolleri

MS SQL sunucuda bir giriş/çıkış(I/O) hatasıyla karşılaşıldığında bu durum sunucunun hata günlüklerine düşer ve sunucu ile iletişim kesilir. Bir sayfa üzerinde okuma yapılırken bir hata alınması durumunda ise bu hata bilgisi MS SQL VT üzerinde bulunan şüpheli sayfalar (*suspect_pages*) tablosuna kaydedilir. Bozulma durumu düzeltildikten sonra bu tablonun silinmesi(*truncate*) gerçekleştirilmelidir. Böylece daha önceden karşılaşılan ve düzeltilen sayfaların tekrar tekrar ortaya çıkması ve bunun bir karışıklığa sebebiyet vermesi önlenmiş olur. Ayrıca bu tür bir hatanın oluşması durumunda SQL sunucu üzerindeki ajan (*agent*) üzerinde bir alarm kurgulamak ve ilgili sorumlulara e-posta, SMS gibi çeşitli kanallardan bilgilendirme yapılması karşılaşılan bir hatada erken aksiyon alma imkânı sunar.

2.2.2 Sayfa Koruma Seçenekleri

MS SQL sunucu motoru kontrol mekanizmaları sayesinde sistem tarafından kopuk sayfa algılama, sayfa kontrolü ve otomatik sayfa kurtarma şeklinde üç tür hata tespitinde bulunabilir. Bunların hangisine izin verilip verilmemesi VT yöneticilerinin tercihine bırakılmıştır.

Kopuk Sayfa Algılama (*Torn-Page Detection*): MS SQL mimarisinde bir sayfa (*page*) 16*512= bit olarak hesaplanır yani 8 KB'dır.

Bir VT üzerinde yer alan sayfalara ait örnek liste görüntüsü Şekil 2'de gösterilmektedir.

Results | Messages

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	5	1647	NULL	NULL	1028198713	1	1	72057594049134592	In-row data	10	NULL	1	1599	0	0
2	5	1646	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	5	512527	4	282555
3	1	347633	5	1647	1028198713	1	1	72057594049134592	In-row data	2	1	4	497115	0	0
4	1	347634	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	3	369267	3	450368
5	4	56034	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	3	356327	3	314914
6	1	1376	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	4	279559	4	278755
7	4	56055	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	5	409196	4	302242
8	1	1486	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	4	278759	3	471166
9	1	1596	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	4	320333	1	505782
10	5	404736	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	4	314014	4	285703
11	5	404737	5	1647	1028198713	1	1	72057594049134592	In-row data	1	0	3	335884	5	404743

Query executed successfully. 00:00:00 10377 rows

Şekil 2. VT sayfaları liste örneği.

Her bir sayfa bilgisinin yer aldığı satırda o sayfadan önceki ve sonraki sayfalara ait sayfa ID (*PrevPageFID ve PrevPagePID*) bilgileri yer alır. DBCC IND söz dizimi [16] kullanımı sonucunda bir tablo veya dizinin kullandığı sayfanın dosya kimliği(*PageFID*), dosyadaki sayfa numarası(*PagePID*), nesne(*ObjectID*), dizin(*IndexID*) gibi veriler listelenir.

Bu sayfaların yazım esnasında yani 512 bit'lik serinin bir kısmı diske yazılırken VT erişilemez duruma gelirse yazılmaya çalışılan dosya üzerinde bir takım eksiklikler oluşur. Bu durumda kopuk veya yırtık sayfa olarak da adlandırılan eksik yazma işlemi tespit edilir. Ancak bu özellik eski bir algoritma olup fiziksel bozulmaları ve sayfa ortasındaki bozulmaları tespit edemez. MS SQL 2005 sonrası sürümlerde daha gelişmiş bir özellik olan sayfa kontrolü(*page checksum*) kullanılması önerilmektedir. Torn-Page Detection'ı aktif etmek için '*ALTER DATABASE Ornek_DB SET PAGE_VERIFY TORN_PAGE_DETECTION WITH NO_WAIT'* örnek komut seti kullanılabilir.

Sayfa Kontrolü (*Page Checksum*): MS SQL'de sayfa bazında toplu doğrulama yapma algoritmasına dayanır. TempDB dışında tüm veri tabanları oluşturulduğunda sayfa kontrol parametresi varsayılan olarak seçili gelmektedir. Sayfa başına hesaplanan 4 byte'lık bir değer sayfa başlık (*page header*) bilgisinde tutulur. SQL sunucusu diske bir değer yazacağı zaman bu hesaplamayı yapar ve bunu başlık bilgisine kaydeder. Bu sayfa diskten yeniden okunurken bu değer yeniden hesaplanır. Başlık bilgisine yazma sırasında önceden yazılmış olan tuttuğu veri ile karşılaştırması sonucunda bu değerler birbiri ile aynı ise sürece devam eder. Aksi halde sayfanın bozuk olduğuna karar vererek hata üretir. VT'nın sayfa doğrulama(*page verify*) özelliğini el yordamıyla değiştirme için '*ALTER DATABASE Veritabani_ADI DB SET PAGE_VERIFY CHECKSUM WITH NO_WAIT*' sözdizimi örneği kullanılır.

Otomatik Sayfa Kurtarma (A*utomatic Page Repair*): Aynalama gibi güncel veri yedekleme teknolojilerinde çalışan birincil veya ikincil sunucuda sayfaların otomatik olarak düzeltilmesini sağlayan bir teknolojidir. Otomatik sayfa kurtarma; "*824-Soft I/O Error*", "*823-Hard I/O Error*" ve "*829-In Restore*" hatalarını düzeltebilir. Bu düzeltme asenkron olarak ilgili sayfa okunduğu veya yazıldığı durumlarda gerçekleşir. Çevrimiçi hata oluşmaz.

2.3 Bozulma Tutarlılık Kontrolleri

VT ve yedek dosyalar üzerinde iki temel tutarlılık kontrolü kullanılır. VT üzerinden sayfa, dizin, sistem tablosu, bilgi tutarlılıkları ile fiziksel ve mantıksal tutarlılık kontrolü gerçekleştirilir. Disk, sürücü ve bellek gibi diğer donanım kaynaklı bozulmalara ilişkin tutarlılık kontrolü ise yedek dosya üzerinden yapılır.

2.3.1 Veri Tabanı Kontrolü

Kopuk sayfa algılama ve sayfa kontrol yöntemlerinde bir sayfanın diske yazımı sırasında oluşan bir hatadan dolayı, diske yazamadığı durum hakkında bilgi vermektedir. Oysa veri tabanı üzerindeki yazma işlemi yapılmayan sayfalar haricindeki bölümlerde oluşan bir takım bozulmaların tespitinde bu yöntemler kullanılamaz.

MS SQL VT üzerindeki dosyaların tutarlılık kontrolleri için VT konsol komutlarından DBCC 'CHECKDB' kullanılır. DBCC CHECKDB VT üzerinde gerçekleştirilen ana tutarlılık kontrolüdür. Bu kontrol işleminde veri tabanı üzerindeki tüm dosyalar sırayla okunur, bu okuma işlemi sırasında okunan her bir dosyanın sistem tarafından daha önce belirlenen sırada olup olmadığı kontrol edilir. VT isimlendirme kurallarına uygun şekilde isimlendirme yapılır. Bir isim yazılmadığında veya '0' değeri girildiğinde çalışılan geçerli veri dikkate alınır.

DBCC CHECKDB sözdizimi Şekil 3'de, kullanılan parametreler ve kullanım amaçları [17]:

```
DBCC CHECKDB
    [ ( database_name | database_id | 0
        [ , NOINDEX
        | , { REPAIR_ALLOW_DATA_LOSS | REPAIR_FAST | REPAIR_REBUILD } ]
    ) ]
    [ WITH
        {
            [ ALL_ERRORMSGS ]
            [ , EXTENDED_LOGICAL_CHECKS ]
            [ , NO_INFOMSGS ]
            [ , TABLOCK ]
            [ , ESTIMATEONLY ]
            [ , { PHYSICAL_ONLY | DATA_PURITY } ]
            [ , MAXDOP  = number_of_processors ]
        }
    ]
]
```

Şekil 3.MS SQL VT kontrol söz dizimi

NOINDEX. Kullanıcı tabloları üzerindeki kümelenmiş dizin(*clustered index*)'lerin kontrol edilmemesini sağlar. Bu işlem genel çalışma süresini kısaltır.

REPAIR_ALLOW_DATA_LOSS. DBCC CHECKDB tarafından rapor edilen hataların onarılması için kullanılır. VT'nın tek kullanıcı modunda olması gerekmektedir.

REPAIR_FAST. Söz dizimi yalnızca geriye dönük tutarlılık için korunur. Hiçbir onarım işlemi gerçekleştirilmez.

ALL_ERRORMSGS. Tüm hata mesajları varsayılan olarak görüntülenir. Bu seçeneği belirtip belirtmemenin hiçbir etkisi yoktur.

REPAIR_REBUILD. Veri kaybı olasılığı olmayan onarımları gerçekleştirir. Kümelenmiş dizin üzerindeki eksik satırların onarımı veya yeniden dizin oluşturma (*index rebuild*) türü işlemlerin gerçekleştirilmesini sağlar. FILESTREAM verilerini onarabilme yeteneği yoktur. Bu durumda varsa hataları onarmak için bir yedekten geri yükleme önerilmektedir.

EXTENDED_LOGICAL_CHECKS. Uyumluluk seviyesi SQL 2008 sonrası ise XML '*index*' ve '*spatial index*' üzerinde tutarlılık kontrolü yapar.

NO_INFOMSGS. Hatalar dışında üretilen tüm bilgi denetim iz kayıtlarını baskılar.

TABLOCK. İşlem yapılan VT üzerinde anlık ekran görüntüsü(*snapshot*) almak yerine tablo üzerine kilit(*lock*) koyarak ilerler. DBCC CHECKDB işleminin daha hızlı çalışmasını sağlar.

ESTIMATEONLY. Gerçek VT kontrolü yapmadan, DBCC CHECKDB işlemi için ne kadarlık TempDB alanına ihtiyaç duyulacağını hesaplar.

PHYSICAL_ONLY. Sayfaların fiziksel yapısının bütünlüğünü ve tahsis edilen alan tutarlılığını denetler. Beraberinde '*Torn-Page Detection' ve 'Checksum*' denetimi yapar. VT fiziksel tutarlılığının daha hızlı kontrol edilmesini amaçlar.

DBCC CHECKDB ana tutarlılık kontrolü yapılır. DBCC IND komutuyla bir tablo veya index üzerindeki tüm sayfaların listelenmesi sağlanır. Söz dizim yapısı [18]: *DBCC IND ({Veritabanı_ADI | dbid}, {objectname | objectid}, {Nonclustered index id |1 | 0| -1 | -2}, [partition_number]).*

Bir sayfanın içeriğini görmek için ise DBCC PAGE komutu kullanır. Öncesinde *DBCC TRACEON(3604);* söz dizimi çalıştırılmalıdır. DBCC PAGE söz dizim yapısı: *DBCC PAGE ({'Veritabanı_ADI | dbid'}, filenum, pagenum [, printopt= {0|1|2|3}]) WITH TABLE RESULT*. Kullanım örneği; *DBCC PAGE('OrnekDB', 1, 404736, 3) WITH TABLE RESULT*

DBCC DBINFO komutuyla VT'na ait 'boot page' bilgisini gösterir. *DBCC DBINFO [('dbname')]*

Hassas Sistem Tablolarının Kontrolü: Bu tablolar üzerinde karşılaşılan bir sorun durumunda alınan hataya ilişkin otomatik onarma('restore') işlemi gerçekleşemez. Bu işlemin el yordamıyla gerçekleştirilmesi gereklidir.

Tahsis Durum Kontrolü: Sayfa türü veya sahip olduğu nesne türüne bakmaksızın VT'ndaki bütün sayfaların tahsis durumlarının kontrolünü yapar. Yine bu sayfalar aralarındaki ilişkileri takip etmede iç yapılanmaları kontrol eder.

Tablo Tutarlılık Kontrolü: Hassas/kritik sistem tabloları ve tüm VT'na ait tabloların fiziksel ve mantıksal tutarlılık kontrolleri gerçekleştirir.

Üst Veri Kontrolü: VT'na ait üst veri(*metadata*) kontrolü yapar.

Indexed View, XML Index, Spatial Index Kontrolü: Dizin verilerinin tutarlılığını kontrol eder.

2.3.2 Yedek Dosyası Kontrolü

Yedek dosya kontrolü (*backup checksum*), yedekleme işlemi sırasında VT üzerindeki tüm sayfalar üzerinde, sayfa kontrol işlemini gerçekleştirir. Bu doğrulamama işlemi sırasında bir sorun bulunması durumunda yedek alma işlemini durdurarak hata mesajıyla kullanıcıya soruna ilişkin bilgiler sunar. Ayrıca tüm yedekler üzerinde; yedek zincirini bozmayacak tam, fark ve günlük yedekleri içinde bir tutarlılık kontrolü gerçekleştirir ve bunu yedek başlığı üzerine kaydeder. Böylece onarmanın hangi yedek sonra ne sürede yapılabileceğine dair bilgiye ulaşılır. Yalnız bu işlem disk üzerine aşırı bir yük oluşturacaktır. Bu sebep ile yedek alma işlemi normalde sürdüğünden daha uzun ve daha maliyetli olacağının da unutulmaması gerekir. Eğer yedek bu parametre ile alınır ise sonrasında '*RESTORE VERIFYONLY*' seçeneği ile alınan yedekler üzerinde bir tutarlılık kontrolü yapılmasına izin verir.

3 Veri Tabanı Kurtarma Yöntemleri

VT kurtarma(*database recovery*), bozulan bir VT'nı en az veri kaybıyla tekrar çalışır duruma getirmek için yürütülen geri yükleme işlemlerini içerir. Veri kurtarma sırasında mevcut veri tabanının çalıştırılamaması veri hacmi ve işlem yoğunluğu yüksek olan veri tabanlarında işlem kesintilerine sebep olabilir. Bu durumda kurtarma dönüşüm hızına bağlı olarak çevrim içi sistemlerde veri kayıplarının yaşanması söz konusudur. Planlı ve kuruma uygun biçimde yapılandırılmamış bir VT kurtarma çalışması disk kullanımının icrasını olumsuz etkiler.

Kurtarma modeli, MS SQL VT'nın kurtarılmasında kullanılacak VT işlem günlüklerinin nasıl bir biçimde tutulması gerektiğine dair seçimdir. MS SQL VT kurtarma teknikleri basit(*simple*), tam(*full*) ve toplu kayıtlı(*bulk-logged*) olmak üzere üç ayrı modele sahiptir. Bu modeller yedekleme, geri yükleme süreci ve kurtarma prosedürü bakımından seçilen kurtarma model ve parametrelerine göre değişiklik gösterir.

SQL Server içerisinde belirlenen kurtarma modeli ayarıyla VT motorunun işlem günlüğüne ne kadar veri yazması gerektiğini, yedekleme ve belirlenen bir zaman noktasına ait geri yükleme seçenekleri belirlenir. VT kurtarma modeli ayarı, '*SQL Server Management Studio'*

uygulama üzerinden veya '*Microsoft Transact-SQL'* gibi ara yüzler üzerinden çalıştırılabilir. VT kurtarma model değişimine ilişkin söz dizimi örneği Şekil 4'de verilmiştir.

Şekil 4. VT kurtarma model değiştirme sözdizimi örneği.

Bu örnekte VT kurtarma modeli tam('FULL') olarak değiştirilmiştir. Diğer kurtarma modellerine geçiş için aynı söz dizimde 'SET RECOVERY' parametresinden sonra tercih edilecek modelin ismi yazılarak kullanılabilir.

3.1 Basit Kurtarma

Kurtarma yöntemlerinin en temel ve sade olanıdır. VT kurtarma modeli basit olarak ayarlanması durumunda işlem günlükleri, son yapılan işlemler, program hata çıktı kayıtları tutulmaz ve kurtarma işleminde kullanılamaz. Bir bozulma/hata yaşanması durumunda en son alınan yedekten(*backup*) sonraki tüm veri kaybedilir. Bu nedenle genelde test, geliştirme ve raporlama ortamlarında kullanılır. Bu kurtarma yönteminde sırayla tam yedek ve varsa son fark yedekleme dosyası geri yüklenir. Kullanımı kolaydır, üst seviye bilgiye ihtiyaç duymaz, veri değişmez.

3.2 Tam Kurtarma

Tam kurtarma modeli, tüm yedekleme dosyalarının kullanılabildiği eksiksiz veri kurtarmada etkili olan bir kurtarma modelidir. Bu modelde VT üzerinde gerçekleşen aktivitelere ait kayıtlar işlem günlüğüne yazılır ve bu kayıtlar işlem günlüğünün yedeklemesi yapılana kadar bu günlüklerinde tutulmaya devam eder. İşlem günlüğü yedeklemesi yapıldıktan sonra günlükten silinir. İşlem günlüğü kapasitesi, yedekleme süresince oluşacak her türlü işlem, dizin oluşturma, değiştirme gibi günlüğe yazılan tüm aktiviteleri depolayabilecek yeterlilikte olmalıdır. Aksi halde işlem günlüğünü dolar. VT işlemleri kabul etmeyi durdurarak yeni kayıtların işlem günlüğüne yazılması engellenir. Bu tür bir olumsuz sonuçla karşılaşmamak için bu modelin seçildiği durumlarda işlem günlüğü yedeklerinin işlem hacim ve kapasitesi dikkate alınmalı ve bu durum izlenmelidir. İşlem günlüğü kayıtlarının tam alınabildiği bir tam kurtarma modelinde her hangi bir t anında yaşanabilecek bir VT bozulma durumu yaşanma halinde işlem günlüğü kayıtlarından her bir işleme ait işlem kaydına erişebilmek mümkündür. Bu model aynı zamanda VT yedekleme ve geri yüklemeye ait tüm seçenekleri destekler. Özetle tam kurtarma modelinde işlem günlüğü kayıtları üzerinden VT istenilen belirli bir t anına dönüş yapılır. Bu modelin kullanılmasıyla yapılacak bir geri yükle sonucunda veri kaybının en az yaşandığı durumdur. Yüksek erişilebilirlik gerektiren kritik iş süreçlerinde kullanılır. Bu modelin kullanımında geri yükleme işlem sırası aşağıda sıralanmıştır.

- İşlem günlüğü yedeği alınır,
- Tam yedekleme dosyaları geri yüklenir.
- Son fark yedekleme dosyaları geri yüklenir.
- Tüm işlem günlüğü yedekleri sırayla geri yüklenir.
- Varsa son işlem günlüğü yedeği geri yüklenir.

3.3 Toplu Girişli Kurtarma Modeli

Toplu olarak günlüğe kaydedilen (*bulk-logged*), kurtarma modelinde "*BULK INSERT, SELECT INTO"* veya *"CREATE INDEX*" gibi toplu işlemlerin günlüğe kaydedilmesinde işlem günlüğü veri alanı kullanımı en düşük düzeyde tutularak günlük hacmi daha küçük olur ve toplu kopyalamadan kaynaklı dolayı yüksek performans gösterir. Bunların haricinde tam kurtarma modeline benzer bir işlev görür. Minimum günlük kaydı, çok fazla bilgi kaydetmeyerek günlüğün daha küçük tutulmasına yardımcı olur. Aynı zamanda işlem günlüğü kapasite bakımından daha etkin kullanılarak zamanla işlem günlüğünde yer kalmama riskini azaltan bir durumdur. Ancak toplu olarak günlüğe kaydedilen işlemlerin günlüğe en az düzeyde kaydedilmesinden dolayı belirli bir noktada yaşanan bozulmadan kaynaklı VT kurtarmaları olumsuz etkilemesi söz konusudur. Bu yöntemle günlüğe kaydetme işlemi gerçekleştirilirken VT'nı yapısının zarar görmesi durumunda yalnızca ilk toplu günlüğe kaydetme işleminden önce oluşturulan son işlem günlüğü yedeği kurtarılabilir. İşlem günlüğü yedeğinin toplu olarak günlüğe kaydeden bir işlem içermesi durumunda ise VT durdurma seçenekleri kullanılamaz. Bu modelin kullanımında toplu olarak günlüğe kaydedilen bir işlemin gerçekleştirilmediği durumunda ise tam kurtarma modelinde olduğu gibi ancak yedeklemenin sonuna kadar olan yerler kurtarılabilir. Bu nedenle, toplu yükleme işlemlerini kullanırken erişilemeyen veri miktarını en aza indirmek için, toplu yükleme işlemi öncesinde ve yükleme sonrasında işlem günlüğü yedeklerinin alınması büyük önem taşır. Böylece toplu yükleme işleminden önce alınan herhangi bir işlem günlüğü yedeklemesinin yanı sıra, işlemin tamamlanmasını takiben özel günlük yedeklemesi alındıktan sonra alınan herhangi bir işlem günlüğü yedeklemesi kullanılarak bir belirli nokta kurtarma işleminin gerçekleştirilebilmesi mümkündür.

Özetle bu model toplu işlemler, VT değişiklikleri ve dizin yeniden oluşturmada en az günlük kaydı içeren bir nevi tam kurtarma modeli gibidir. Günlüğün zarar görmesi veya en son günlük yedeklemesinden sonra bir takım toplu işlemler yapılması söz konusu ise bu durumda yedekleme sonrası gerçekleştirilen işlemlerin yeniden yapılması gerekir. Aksi durumda her hangi bir kayıp yaşanmaz.

3.4 Kurtarma Model Seçimi

Kurtarma model seçiminde basit, tam veya toplu kayıt kurtarma yöntemlerinden birisi seçilebilir, bu seçim VT çalışırken değiştirilebilir. Bu yöntemlerin seçiminde iş gereksinimleri, düzenleyici kurum düzenlemeleri, yasal uyumdan kaynaklı gereklilikler, VT yönetiminde dış kaynak kullanımından kaynaklı durumlar, yedekleme ve geri yüklemede veri bütünlüğüne duyulan ihtiyaç gibi etkenler rol oynamaktadır. Kurum yöneticileri veya işin teknik uzmanları tarafından bu ve benzeri gereklilikler göz önünde tutarak ihtiyaçlarına özgü belirleyecekleri kurallara uygun olarak işletilmesini sağlamalıdır. Sektörden edindiğimiz uzman görüşlerine göre pratik uygulamada yaygın olarak genelde tam veya basit kurtarma modelleri tercih edilmektedir.

Tam kurtarma modeli genelde hassas veri içeren, bir bozulma durumunda en az veri kaybı yaşatma ihtimali olan

ancak modelin kullanımında üst seviye teknik bilgi gerektiren, bir VT uzman veya firma desteğine ihtiyaç duyan, yedekleme ve geri yükleme işlemleri uzun zaman alan bir yöntemdir. Bütün VT'nın geri yüklenmesine ait bir senaryoda öncelikle tam bir VT yedeği, ardından varsa fark yedeği ve daha sonra tüm günlük yedeklerin sırayla art arda geri yüklenme adımlarını içerir.

Dosya geri yüklemeleri genelde VT çevrimdışıyken veya SQL Server'ın bazı sürümlerinde ise VT çevrimiçi içindeyken gerçekleştirilebilir. Bir veya daha fazla hasarlı sayfanın geri yüklenmesinde benzer yapı kullanılabilir.

Parça parça geri yüklemede VT'nı, birincil dosya grubundan başlayarak dosya grubu düzeyinde aşamalar halinde geri yükleme yapılarak kurtarma işlemi gerçekleştirilir. Bu yöntemin uygulanmasında ayrı bir sunucuda kapsamlı bir teste tabi tutmak önerilmektedir.

Bir VT için kullanılabilen geri yükleme işlemleri, kurtarma modeline bağlıdır. Microsoft SQL sunucu VT kurtarma modellerinin hangi senaryoyu ne ölçüde destekleyip desteklemediğine dair Microsoft firmasının kendi dokümanından yararlanılarak oluşturulan özet bilgiler Tablo 1'de yer almaktadır [19].

Tablo 1. Kurtarma modeli ve desteklenen geri yükleme işlemleri.

Geri Yükleme İşlemi	Basit Kurtarma Modeli	Toplu Kurtarma Modeli	Tam Kurtarma Modeli
Veri Kurtarma	En son tam ve fark yedekten sonraki tüm veriler kaybolur.	Bazı verilerin kaybı gerçekleşebilir.	Günlük varsa tam kurtarma
Anlık geri yükleme	Desteklenmez	Günlük yedekleme, toplu olarak günlüğe kaydedilmiş değişiklikler içeriyorsa buna izin verilmez.	Günlük yedeklemeleri kapsamındaki herhangi bir zamanı destekler.
Dosya geri yükleme	Yalnızca salt okunur ikincil dosyalar için kullanılabilir.	Kısmen	Tam destek
Sayfa geri yükleme	Desteklenmez	Kısmen	Tam destek
Parça parça (dosya grubu düzeyinde) geri yükleme	Yalnızca salt okunur ikincil dosyalar için kullanılabilir.	Kısmen	Tam destek

Günlüğü dolduran ekleme ve güncelleme işlemlerine ek olarak, dizin oluşturma/değiştirme ve toplu yükleme işlemleri gibi diğer işlemler işlem günlüğüne birçok veri kaydeder. İndeks ve toplu yükleme işlemleri (SELECT INTO gibi) nedeniyle işlem günlüğünün dolma riski varsa bu işlemler gerçekleştirilirken toplu olarak günlüğe kaydedilen kurtarma modeli tercih edilir. Diğer açılardan tam kurtarma modeline benzemektedir.

Kritik olmayan sistemler için veya tam kurtarma modelini kullanabilme yetkinliğine sahip olunmaması gibi durumlarda basit yöntemin tercih edilmesi mümkündür. Bu modelde işlem günlüklerine ihtiyaç duyulmadığı için günlük alınan bir VT yedeği yeterlidir. Bazı durumlarda, veri kaydının kaynak sistemden depolanması ve iletiminde önbellek üzerinde verinin bulunması ve o anda elektrik kesintisi gibi bir sorunla karşılaşılması durumunda bu verilere erişim sağlanamama durumu oluşabilir. Bu gibi nedenlerle kurtarma işleminin SQL Server sistemi çevrimdışındayken yapılması veri kaybı riskini azaltır. Sürekli büyüyen işlem günlüklerine ve ileri teknik uzmanlık bilgisine ihtiyaç duymaması bir avantajken sayfa geri yüklemeyi desteklemiyor olması ve veri kaybının yaşanma ihtimalinin yüksekliği olumsuz yanıdır.

Bir VT'nın geri yüklenmesinde VT motoru temel üç adımı uygular. VT bütünüyle bozuksa öncelikle VT ve işlem günlüğü dosyalarını oluşturur. Ardından VT yedekleme ortamından tüm verileri, günlükleri ve dizin sayfalarını VT dosyalarına kopyalar. Son olarak kurtarma işlemi olarak bilinen işlem günlüğünü uygular. Dosya geri yüklenmesinde ise; eksik VT dosyaları oluşturulur ve ardından yedek ünitelerinden ilgili VT dosyaları kopyalanır. Diğer taraftan VT, dosya, sayfa, parça parça geri yükleme senaryolarının kurtarma yöntemleriyle eşlenmesine dair özetlenen bilgiler Tablo 2'de yer almaktadır. Bu veriler Microsoft firması tarafından sunulan bilgilerden yararlanılarak hazırlanmıştır [19].

Tablo 2. Geri yükleme senaryolarına göre yöntemlerin kullanımı.

Senaryo	Basit Kurtarma Yöntemi	Tam/Toplu Kurtarma Yöntemi
VT geri yükleme	VT tam yedeği yüklenir, ardından fark yedeği geri yüklenir.	VT tam yedeği, ardından varsa son fark yedeği, daha sonra sırayla tüm günlük yedekler geri yüklenir.
Dosya geri yükleme	Tüm VT geri yüklenmeden hasarlı salt okunur dosya geri yüklenir.	Dosya geri yükleme, VT çevrimdışı/bazı sürümlerde çevrimiçi iken yapılabilir.
Sayfa geri yükleme	Uygulanamaz	Hasarlı sayfalar geri yüklenir. Geri yüklenmekte olan sayfalar her zaman çevrimdışıdır.
Parça parça geri yükleme	VT, birincil ve tüm okuma/yazma, ikincil dosya grubu düzeyinden başlanarak gerçekleştirilir.	VT, birincil dosya grubundan başlayarak dosya grubu düzeyinde aşamalar halinde geri yüklenir ve kurtarılır.

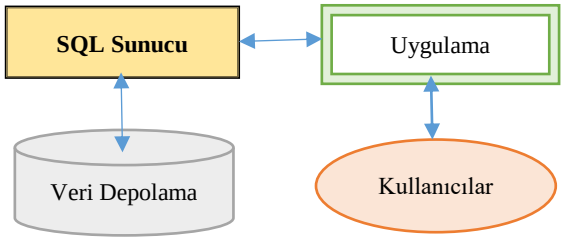
4 Uygulama

Gelecekte bir t anında yaşanması muhtemel bir VT bozulma olayı düşünülerek VT sistemlerinin planlaması ve kurulumunda teknoloji, mimari, kurtarma model seçimi ve test önemlidir. Bozulma olayı sonrasında daha etkin sonuçlar üretebilecek doğru yaklaşımlar tercih edilmelidir. Bir olay yaşanmadan önce kurulan yapının kontrol ve testleri çeşitli senaryolar üzerinden çalışılmalıdır. Bu uygulamalı çalışma, bir bozulma sonrası veri kaybı ve veri erişim problemlerinin en kısa sürede en az hasarla çözülmesine destek olur. Bu bölüm VT mimari ve kurtarma senaryolarının oluşturulması ile mevcut durumunu gözlenmesine uygulama pratiğine katkı sunar.

4.1 Fiziksel Mimariye Dayalı Mimariler

Veri kurtarmada, VT kurtarma modellerinin seçimi kadar SQL sunucu sistem mimarisinin türü ve kurulumu önemlidir. Bu bölümde fiziksel cihazlar kullanılarak modellenen VT mimari yapıları örnek senaryolar üzerinden aşamalı olarak sunulmaktadır. En temel bir

mimari yapıda bir SQL sunucusu(*SQL server*) ve ona bağlı bir veri depolama birimi(*storage*) bulunur. Kullanıcılar bir uygulama ara yüzü kullanarak sunucu üzerinden veri depolama birimi üzerindeki verilere ulaşabilirler. Bu temel VT mimari yapı örneği Şekil 5'de sunulmaktadır.

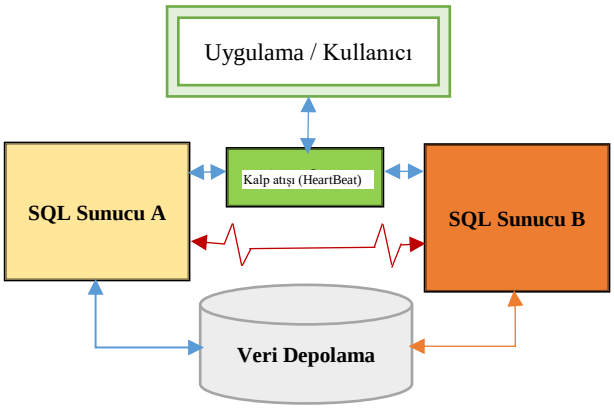


Şekil 5. Temel VT mimari örneği.

Bu temel mimaride sunucu üzerinde oluşacak yazılım ve donanım kaynaklı bir hatadan dolayı yaşanacak bir bozulma durumunda kullanıcıların veri depolama birimi ile iletişimi kesilir. Bu veri okuma, yazma, güncelleme gibi veri işlemleri kesintiye uğratır. SQL sunucuda oluşan hata ve bozulma nedenine bağlı olarak sistemin yeniden çalışır hale getirilmesi uzun zaman alabilir. Bu yapı donanım ve kurulum maliyeti gibi bir takım avantajlara sahiptir. Ancak veri kaybı ve erişimine tahammülü olmayan kurumlar için istenmeyen, tercih edilmeyen bir mimari yapıdır.

4.1.1 SQL Sunucu 'Fail Over Cluster' Yapısı

Sunulan örnek temel mimaride SQL sunucusunda yaşanabilecek bir bozulmada bu sistemin yeniden çalışır hale gelebilmesi için yeni bir sunucunun temin edilerek devreye alınmasına bile gerek duyulabilir. SQL sunucunun çalışmaması demek veri iletişiminde kesintiye uğraması demektir. Bu tür sunucu hatalarında veri iletişim sorununu çözmede mevcut SQL sunucusu yanına yedekli çalışabilecek bir yapıda ilk etapta 'pasif' olan ancak bir bozulma durumunda 'aktif' hale getirilerek veri iletişim kaybını azaltan bir model oluşturulabilir. Buna ilişkin örnek mimari yapısına teknik olarak '*fail over cluster*' mimarisi olarak isimlendirilmektedir. Bu mimari yapı örneği Şekil 6'da gösterilmektedir.

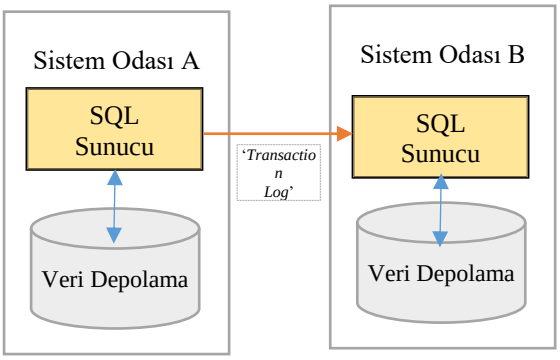


Şekil 6. 'Fail Over Cluster' mimari örneği.

Birden fazla SQL sunucu ve veri depolama ünitesinin birlikte çalıştığı yapıya kümeleme(*cluster*) denilmektedir. Bu yapı üzerindeki SQL sunuculardan biri üzerinde yaşanacak bir hata(*fail*) durumunda diğer cihazın devreye alınarak sistemin az kayıpla çalışması mantığına dayanır. Bu örnek yapıda ilk etapta A sunucusu aktif iken B sunucusu pasif konumlandırılmıştır. Sunucular iki sunucu arasındaki kalp atışı(*HeartBeat*) adı verilen özel bir ağ aracılığıyla birleriyle sürekli bir haberleşme gerçekleştirir. Örneğin A sunucusunda bir hata oluşması durumunda, B sunucusu bu iletişim sayesinde haberdar olur. Hemen veri depolama birimi ile bağlantıyı kurar ve veri depolama ünitesindeki disk ünitelerini aktif hale getirir, sanal IP'yi kendi üzerine alır, SQL servis hizmetini çalıştırır. Böylece uygulamalar üzerinden gelen kullanıcılar sanal IP üzerinde tanımlı olan B SQL sunucusunu kullanarak veri depolama birimi ile iletişimi devam ettirirler. Bu yapı bir önceki temel yapıda belirtilen sorunu çözen bir yapıdır. Yani sunucu bozulmaları için yedekli bir yapıda veri iletişim kesintisi düşük seviyede gerçekleşir. Ancak bozulma sunucular yerine veri depolama cihazında veya cihazların bir arada bulunduğu sistem odasında gerçekleşmesi durumunda bu yapı yetersiz kalmaktadır. Bu yetersizliği gidermek amacıyla farklı yaklaşımlar geliştirilmiştir.

4.1.2 'Log Shipping' Yapısı

SQL sunucu ve veri depolama cihazlarının tek bir konumda yer almasından kaynaklı sorunu çözmek için geliştirilen bir yöntemdir. Bu yöntemde farklı konumlarda yer alan iki sistem odasındaki SQL sunucu sistemlerinin birinde çevrimiçi işlemler gerçekleşirken bu işlemler diğer sistem odasındaki SQL sunucu üzerinde belirli bir zaman diliminden sonra işlem kayıtlarının(*transaction log*) kopyalanması şeklinde gerçekleşmektedir. Bu mimari örneği Şekil 7'de gösterilmektedir.

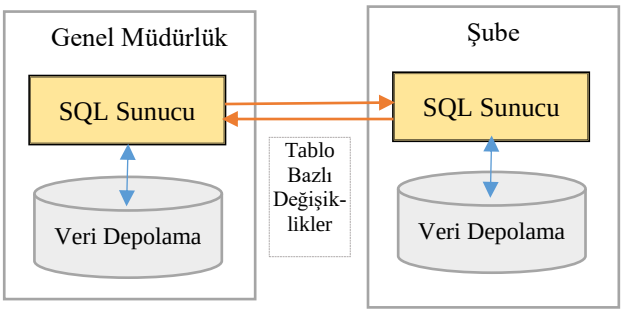


Şekil 7. 'Log Shipping' mimari örneği.

Örnek olarak sunulan bu yapıda A ve B sistem odalarında benzer iki sistem modellenmiştir. A odasındaki sistem aktif ve çevrimiçi olarak çalışırken B odasındaki sistemler pasif olarak çalışmaktadır. Yani A odasında gerçekleşen veri işlemleri belirli zaman dilimlerinde B sistem odasındaki sunucu üzerinden veri depolama işlemi gerçekleştirilmektedir. Kısaca aktif çalışan A sistem odasında üretilen verilerin bir kopyası belirli zaman farkıyla B sistem odasındaki yapı üzerine kopyalanmaktadır. Bu yapını olumsuz sayılabilecek taraflarından biri iki tarafında aktif-aktif çalışmaması bir tarafın pasif konumda olması, B sunucu sisteminin etkin kullanılmamasına sebep olmaktadır. Diğeri ise bu yapıda bir '*fail over cluster*' yapısı olmadığından A sistem odasında yaşanacak bir sorunda buradaki kullanıcı, bağlantı IP'si gibi diğer sistem ayarlarının B tarafında manuel yapılmasından kaynaklı iş yükü ve zaman kaybı söz konusudur.

4.1.3 SQL Sunucularının Eşitlenmesi

İki SQL sunucunun eşitlenmesi(*replication*), senkronize olmasında kullanılan yöntemdir. Bu yöntemde iki ayrı sunucu üzerinde tablo temelli değişiklikler karşılıklı olarak bir birlerine belli zaman dilimlerinde iletilir. Bu mimari örneği Şekil 8'de gösterilmektedir.

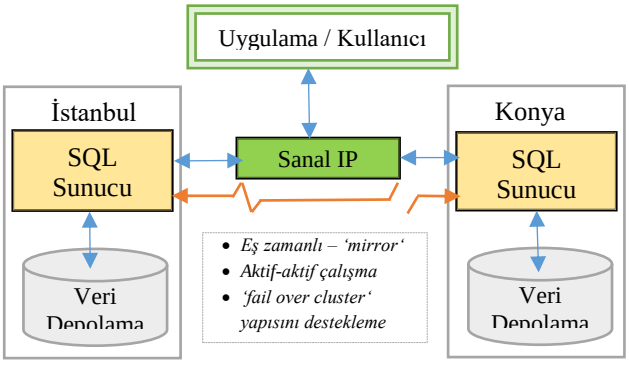


Şekil 8. 'Replication' mimari örneği.

Bu örnekte bir kuruluşun genel müdürlüğü ile uzak bir şubesi arasında kurulan bir yapıyı temsil etmektedir. Bu yapıda gün içinde şubede gerçekleşen işlem kayıtları gün sonu işlemlerinden önce Genel Müdürlük sunucusuna aktarılırken, Genel müdürlük tarafından şubeye iletilecek fiyat değişikliği gibi verilerin ise şube sunucusuna aktarımı gerçekleştirilir. Bu yapıda işlemler çevrimiçi yerine belli bir zaman dilimlerinde gerçekleşmemektedir. Bu yüzden sistemlerin birinde yaşanan sorundan kaynaklı veri kaybı yaşanması söz konusudur.

4.1.4 'Always On' Yapısı

Microsoft 'Always On' teknolojisiyle kurulan örnek model Şekil 9'da gösterilmektedir. Bu örnek mimari yapı, İstanbul'da genel merkezi olan bir kurumun bir felaket durumda erişebilmek için çevrimiçi yedekli yapısını Konya'da kurduğu varsayımıyla üretilmiştir.



Şekil 9. 'Always On' mimari örneği.

Bu yapı önceki anlatılan Microsoft teknoloji ve mimarilerden kaynaklı olumsuz yanları gideren, eşzamanlı aynalama tekniği, sunucu hatalarında '*fail over cluster*' yapısını desteklemesi ve aktif-aktif çalıştırma gibi daha üstün özelliklere sahiptir. İstenirse genel müdürlüğün yer aldığı İstanbul SQL sunucularında VT işlemleri gerçekleştirilirken yedek konumdaki Konya sunucularında iş zekâsı, raporlama ve günlük yedeklerin alınması gibi işlemler yaptırılabilir, iş yükü sunucular arasında paylaştırılabilir. Böylece her iki sunucuyu daha aktif ve verimli kullanma imkânı sağlanmış olur.

4.1.5 Veri Tabanı Aynalama

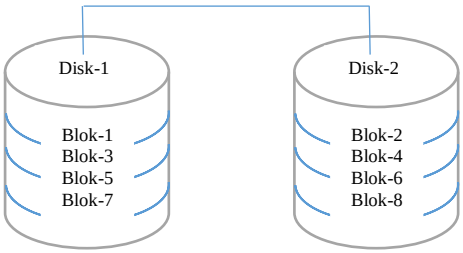
MS SQL VT aynalama (*DB mirror'*) yönteminde iki sunucunun birinde üretilen işlem kaydının otomatik olarak önce yedek sunucuya sonra kopyası kendi üzerine eş zamanlı yazılma işlemidir. Bu yöntemde sunuculardan biri aktif çalışırken diğeri pasif çalışır. Sunuculardan birisinin bozulması durumunda bu yapı, bir 'fail over cluster' yapısında olmamasından kaynaklı veri ve işlem zaman kaybı konusudur.

4.2 RAID Sanallaştırma Mimarisi

RAID, veri yedekliği ve performansın artırımında fiziksel disk sürücüsü bileşenini bir veya birden çok mantıksal birimle birleştiren bir veri depolama sanallaştırma teknolojisidir [20]. VT dosyalarının korunma ve geri dönüşünde uygun bir RAID yedekli yapısının seçimi oldukça önemlidir. Veriler, yedekleme ve performans düzeyine bağlı olarak, RAID sürücüleri arasında dağıtılır.

RAID yapısında sunucunun desteklediği tüm diskler kullanımı mümkündür. Ancak sistemin hızı bu yapıda kullanılan diskler içerisinden en düşük devir hızına sahip diske göre çalışır. Bu yapıda VT dosyaları birden fazla disk üzerinde tutulacağından disklerden herhangi birinde yaşanabilecek bir bozulma gibi bir sorunda VT dosyalarının olumsuz etkilenmesi düşüktür. Ayrıca performans üzerinde pozitif etki sunmaktadır.

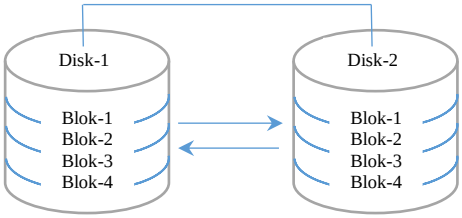
RAID 0: Disk bölüştürme olarak bilinir. Sunucu performansını geliştirmek için kullanılır. Şekil 10'da gösterilen bu yapıda veriler aynı anda birden fazla diskin üzerine yazılır.



Şekil 10. RAID 0 yapısı.

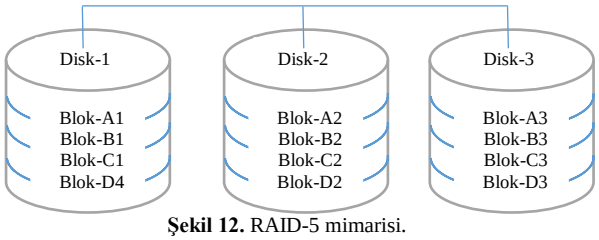
Aynı anda birden fazla diske okuma ve yazma yapabiliyor olması performans artışını sağlar. SQL sunucu tarafında 'temp' VT'nın çalıştırılacak diski RAID-0 yapısında kurgulanabilir. Böylece VT üzerinde disk kaynaklı olarak herhangi bir bozulmanın olması durumunda disk değişimi sonrasında SQL servisleri yeniden başlatılırken 'temp VT'nda yeniden oluşturulacağından sunucu kaldığı yerden çalışmaya devam edecektir.

RAID-1: Bu yapıda iki ayrı diske ihtiyaç duyulur, veriler iki diske aynı anda yazma işlemi yapar. Şekil 11'de örnek bir RAID-1 mimarisi gösterilmektedir. Bu yapıya teknik olarak aynalama denilmektedir. Bu yapıda olumsuz sayılabilecek taraf disk kapasitesinin verimsiz kullanımına ilişkindir. İhtiyacın en az 2 katı diske ihtiyaç vardır. Örneğin 250 GB'lık bir kullanım alanı için disk ünitesinden toplamda 500 GB'lık bir disk alanını ayırmak gerekmektedir.



Şekil 11. RAID-1 yapısı.

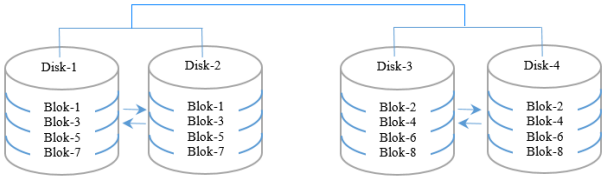
RAID-5: Şekil 12'de sunulan bu yapıda veri üç veya daha fazla disk arasında bölüştürülür. Diskin birinde hata olduğunda dağıtılan hatasız diskte yer alan veri bloğundan otomatik olarak yeniden oluşturulur.



Şekil 12. RAID-5 mimarisi.

Arızalanan disk değiştirilene kadar veri bloğu oluşturulmaya devam eder. RAID-1'e göre daha iyi performans gösterir.

RAID-10: RAID 1+0 olarak da anılmaktadır. RAID 0 ve RAID 1'in bir birleşimidir. Her ikisinin avantajlı yönleri kullanıldığından daha yüksek performansa sahip hataya dayanıklı bir dizidir. Şekil 13'de yer alan bir RAID 10 mimarisi oluşturmak için en az 4 disk sürücüsü gerekir. İki disk aynı anda arızalansa dahi sistem ayakta kalır.

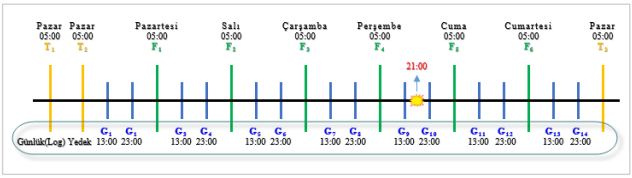


Şekil 13. RAID-5 mimarisi.

İyi performans sunan RAID yapısı olarak bilinir. Pratikte çok fazla yazma işlemi yapan SQL VT sunucularında kullanılması önerilmektedir.

4.3 Veri Tabanı Kurtarma Senaryoları

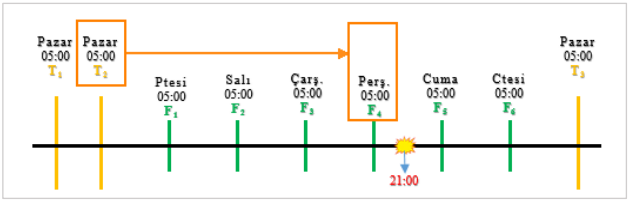
Bu çalışmada VT kurtarmasında tam ve basit kurtarma olma durumlarına uygun bir senaryo kurgulanmıştır. Tam kurtarma modelinin seçildiği senaryoda örnek bir yedekleme yapısı ve bir bozulma anı kurgulandı. Kurgulanan bu model Şekil 14'de gösterilmektedir.



Şekil 14. Tam kurtarmada yedekleme ve bozulma anı kurgusu.

Bu senaryoda 'OrnekVT' isminde bir VT'nında her hafta Pazar günü saat 05:00'de tam yedekleme (T1, T2, T3), diğer günlerde saat 05:00'de ise fark yedekleme (F1, F2, F3, F4, F5, F6) yapıldığı varsayıldı. Yine her gün saat 13:00 ve saat 23:00'de olmak üzere haftada 14 kez (G1, G2, G3, G4, G5, G6, G7, G8, G9, G10, G11, G12, G13, G14) günlük(log) yedeklerinin alınması olarak kurgulandı. Ayrıca VT üzerinde 13.01.2022 Perşembe günü saat 21:00'de bir olay/bozulma gerçekleştiği kabul edilmiştir.

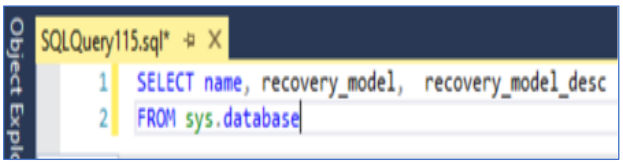
Diğer bir senaryo ise kurtarma yönteminin basit seçilmesi durumu değerlendirilerek tam kurtarma senaryosuna uygun olacak şekilde oluşturuldu. Basit kurtarma yedekleme dönemleri, bozulma anı ve kurtarmada kullanılacak yedeklerin gösterimi Şekil 15'de yer almaktadır.



Şekil 15. Basit kurtarmada yedekleme ve bozulma anı.

Tam ve basit kurtarma senaryolarından da görüleceği üzere bu iki senaryo arasındaki tek fark basit kurtarma modelinde günlük log kayıtlarının alınmaması şeklindedir.

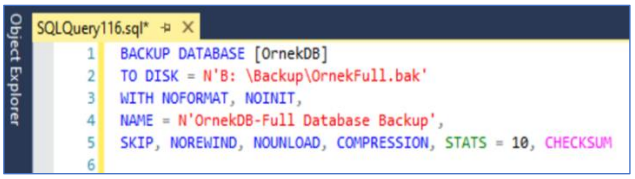
Bu bölümde senaryolarda kurgulanan yedeklemeler ve bozulma olay anı üzerinden VT kurtarma yöntemleri örneklerle çalışıldı. VT'nın hangi modda çalıştığını listelemek için Şekil 16'da yer alan söz dizimi çalıştırılır.



Şekil 16. VT kurtarma modu listeleme.

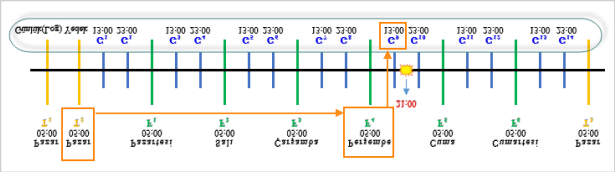
4.3.1 Tam Yedekleme ve Yedekten Dönme

Tam('full') yedekleme, seçilen VT kaynağının tüm içeriğinin yedek alınmaya başlanan andan itibaren yedeği alınır. Bu yöntem veri kurtarmada en güvenilir yöntem olarak bilinmekle birlikte yüksek disk kapasitesine ihtiyaç vardır. Tam yedekleme işlemleri için kullanılacak söz dizimi örneği Şekil 17'de gösterilmektedir.



Şekil 17. Tam yedekleme söz dizimi örneği.

Bu örnek senaryomuzda 13.01.2022 Perşembe saat 21:00'de bir bozulma olayı yaşandığı varsayılmaktadır. Yaşanan bozulma olayı sonrası kurtarma için ilk yapılacak iş en son alınan tam yedekten geri dönme işlemidir. Ardından en son alınan fark yedeğinin döndürülmesi ve en sonunda fark yedekten sonra olay anına kadar alınan günlük yedeklerin dönülmesi işlemidir. Tam kurtarma senaryosu ve kurtarmada kullanılabilecek veri yedekleri temsili olarak Şekil 18'de gösterilmektedir.



Şekil 18.Tam kurtarmada kullanılacak yedek kayıtlar.

Bu senaryoda 09.01.2022 Pazar saat 05:00'de alınan tam yedeği(T2), 13.01.2022 Perşembe alınan fark yedeği(F4) ve olaydan önce alınan günlük log kaydı(G9) gösterecek şekilde kodlanmıştır. Bozulma öncesi alınan en son tam yedekten(T2) dönme söz dizim örneği Şekil 19'da gösterilmektedir.

```
SQLQuery117.sql*
1  RESTORE DATABASE [OrnekDB]
2  FROM DISK = N'B:\Backup\OrnekFull.bak'
3  WITH FILE=1, RECOVERY, NOUNLOAD, STATS= 5,
4  STOPAT = '2022-01-09 05:00:00'
```

Şekil 19. Tam yedekten dönme söz dizimi örneği.

Bu işlem sırasında kullanılacak 'STOPAT' parametresiyle yedekten dönüş yapılacak zaman belirtilir. VT'nı belirlenen bir zaman dilimine almak istendiğinde kullanılır, yoksa kullanılmaz. Belirlenen bir t anından (13.01.2022) önce bu yedeğin alınmış olması gerekmektedir. Kullanılan parametreler ile yedekten geri dönme işlemine son verilip vermeyeceği belirlenir.

4.3.2 Fark Yedekleme ve Yedekten Dönme

Tam yedekten sonra değişen verilere ait yedeklerin alınmasından dolayı fark(*differential*) yedekleme denir. Bu yedeklemede veriler kümülatif olarak alınır. Fark yedeklerindeki veri artış miktarı bir kurumdaki veri işlem hacmine göre değişiklik gösterir. Fark yedeğini almaya başlamadan önce tam yedek alınır, VT'nı kurtarma modeli olarak 'basit' veya 'tam' olarak seçilir. Fark yedeklemeye ait söz dizim örneği Şekil 20'de gösterilmektedir.

```
SQLQuery118.sql*
1  BACKUP DATABASE [OrnekDB]
2  TO DISK = N'B: \Backup\OrnekDB_Diff.bak'
3  WITH DIFFERENTIAL, NOFORMAT, NOINIT,
4  NAME = N'OrnekDB-Diff Database Backup',
5  SKIP, NOREWIND, NOUNLOAD, COMPRESSION,  STATS = 10
```

Şekil 20. Fark yedekleme söz dizim örneği.

Ele alınan senaryoya göre yedekten geri dönüş temel işlemler aşağıdaki sırayla yapılır:

- Bozulan VT üzerinden henüz yedeği alınmamış olan 'tail-log' kayıtlarının yedeği alınır.
- 'WITH NORECOVERY' seçeneğiyle bozulma öncesi 09.01.2022 tarihinde alınan eldeki tam yedeğe (T2'ye) dönülür.
- 'WITH NORECOVERY' seçeneği ile bozulma öncesi (13.01.2022 Perşembe) alınan en son fark yedeğine(F4) dönülür.
- Bozulma sonrası bozulmuş VT'ndan alınan 'tail-log' yedeğinden dönüş yapılarak VT çevrimiçi(*online*) duruma getirilir.

4.3.3 İşlem Günlükleri ve Yedekten Dönme

VT fark yedekleri arasındaki işlem kayıtlarını belirlenen aralıklarla işlem günlükleri(*log*) denilen yedekler üzerinde tutar. VT'nın kullanıldığı yer ve kritiklik düzeyine göre yedek alma süreleri değişkenlik gösterir. Bu seçimlik süre ihtiyaca göre istenilen zaman aralıklarında yapılabilir. Bu sürenin kısa tutulması olası bozulmalarda veri kaybının en az yaşanmasını sağlarken bozulmadan dönüş işlemlerinde ise işlem adet ve süresini artırmaktadır. Senaryoda yer alan günlük yedeklerden G9'a ait yedek alma söz dizim örneği Şekil 21'de gösterilmektedir.

```
SQLQuery7.sql*
1  BACKUP LOG [OrnekDB] TO
2  DISK=N'B:\Backup\OrnekDB_20220113_13_G9.bak'
3  WITH INIT;
4  GO
```

Şekil 21. Log yedek alma söz dizimi örneği.

Çalışmada ele alınan senaryoya göre bozulma öncesi 13.01.2022 Perşembe gününde alınan en son fark yedeğinden sonra saat 13:00'de alınan günlük log yedeğinden(G9) geri dönüş yapılır. Buna ilişkin örnek söz dizimi Şekil 22'de gösterilmektedir.

```
SQLQuery8.sql*
1  USE [master];
2  RESTORE LOG [OrnekDB] FROM
3  DISK = N'B: \Backup\ OrnekDB_20220113_13_G9.bak'
4  WITH FILE=1, NORECOVERY, NOUNLOAD, STAT=5
5  GO
```

Şekil 22. İşlem günlüğü yedeğinden dönme söz dizim örneği.

Bu senaryoda bozulma önce işlem günlüğü sadece G9'dur ancak başka senaryolarda dönülmesi gereken birden fazla günlük olması durumunda eldeki tüm günlükler için benzer geri dönüş işlemleri ayrı ayrı gerçekleştirilir. Bu geri dönüş(*restore*) işlemleri sırasında söz dizim kodu sonuna 'NORECOVERY' seçeneği konularak VT'nın geri dönüş işlemleri yapılırken çevrimiçi çalışması önlenir.

VT kurtarma modunun değiştirilmesine ihtiyaç duyulması halinde Şekil 23'de yer alan söz dizimi kullanır.

```
SQLQuery120.sql*
1  ALTER DATABASE OrnekDB SET RECOVERY FULL
```

Şekil 23. VT kurtarma modu değiştirme söz dizim örneği.

Bu örnekte çalışmada kullanılan OrnekDB VT'nın kurtarma modeli tam(full) olarak set edilmektedir.

5 Sonuç

Veri kurtarma, bütünlüğü bozulan veya silinen verilerin kurtarılma sürecidir. Bu süreç bazen başarılı bir şekilde sonuçlanırken kimi zaman barındırdığı bazı riskler nedeniyle başarısızlıkla sonuçlanabilmektedir. Hatta verinin bütünü veya bir kısmına erişebilmenin hiç mümkün olmadığı durumlarla karşılaşılmaktadır. Veri kurtarma sürecine dair teknik bilgiler yanında taşıdığı zayıflıklar önceden ele alınarak riskleri önleyici iyileştirmelerin ve kullanılan mimariye uygun kurtarma yöntemlerini belirlenmesi ve bunların uygulama testlerinin yapılması

veri kaybından kaynaklı hasarı en aza indirgemede önemlidir. MS SQL VT kurtarma başarısında MDB ve MDF dosyalarının yedeklerinin olması, VT yedeklerinin veri işlem hacmi ve veri kritikliğine göre saat, gün, hafta, ay gibi seçilen belirli dönemlerde alınıyor olması çok önemlidir. Bu süre bankacılık işlemleri, borsa gibi anlık işlemlerin olduğu yerlerde anlık veya birkaç dakika aralıklarla yapılabilir. Bu veri iş ve teknik süreç sahipleri tarafından bilgi güvenliği, iş sürekliliği ve veri analiz çalışmaları sırasında belirlenmesi önerilir.

MS SQL VT'nda bir verinin başarılı olarak kurtarılmasında tam bir yedeklemenin yapılıyor olması, yedekten tam bir dönüş için ise veri erişimi ve güncelleme işlemlerine karşılık gelen güncel bir günlük dosyasına sahip olmaya bağlıdır. Yedekten dönülen verilerin doğruluğunu ters etmek için yeterli disk kapasitesine sahip bir test ortamına dönüş yapılması önerilmektedir. Bu testler, yedekten gerçek ortama dönüşten önce geri yüklenecek verilerde herhangi bir bozulma/silme olup olmadığının önceden anlaşılması ve gerçek ortam verilerinin korunması bakımından kıymetlidir. Kurum ihtiyaçlarına göre en iyi yedekleme ve kurtarma stratejisi her kurumun kendi hedef ve stratejileriyle uyumlu olacak bir şekilde oluşturulmalıdır.

Bu çalışmada, çeşitli nedenlerle hasara uğramış, bozulmuş, silinmiş veya kaybolmuş MS SQL VT verilerinin kurtarılmasında kullanılan yöntemlerin araştırmış, bu tekniklerin örnek senaryolar üzerinden uygulamalı örnekleri oluşturulmuştur. Çalışmada ele alınan yöntem ve uygulama örnekleri ilgili kurum ve paydaşlara sunulmuştur. Benzer çalışmalar yaygın olarak kullanılan My SQL, Oracle, Postgre SQL gibi VT yönetim sistemleri üzerinde de yürütülebilir.

Bilgilendirme

Gerçekleştirilen bu çalışmada Etik Kurul Onay belgesine gerek yoktur. Yazarlar herhangi bir çıkar çatışması olmadığını beyan etmektedir.

Kaynaklar

[1] Barutçugil, İ. (2002). Bilgi Yönetimi. ISBN: 9789758515264, *Kariyer Yayıncılık İletişim,* İstanbul.

[2] Şahinaslan, E., & Şahinaslan, Ö. (2020). Müşteri İlişkileri Yönetimi ve Veri. *Bilgi Çağı'nda Müşteri İlişkileri Yönetimi*, Kriter Yayınevi, pp. 174-196, İstanbul.

[3] Gu, J., Ma, Z., & Hulbert, G. (2001). Quasi-static data recovery for dynamic analyses of structural systems, *Finite Elements in Analysis and Design*, no. 37. 825-841. 10.1016/S0168-874X(01)00070-1.

[4] Kara, Ş. (2012). *Veri Kurtarma Yöntemlerinin Başarılarının Değerlendirilmesi.* (Master's dissertation, Firat University).

[5] Prairie, E. (2020). *39% of Global businesses are not prepared for a ransomware attack Ontrack*. Retrieved October 19, 2020, from https://www.ontrack.com/en-us/press/details/39-of-global-businesses-are-not-prepared-for-a-ransomware-attack.

[6] Son, Y., Kim, M., Kim, S., Yeom, H. Y., Kim N. S., & Han, H. (2020). Design and Implementation of SSD-Assisted Backup and Recovery for Database Systems. *IEEE Transactions on Knowledge and Data Engineering*, cilt 32, no. 2, pp. 260-274

[7] Armoush, A., Kowalewski, S., & Salewski, F. (2008). Recovery Block with Backup Voting: A New Pattern with Extended Representation for Safety Critical Embedded Systems, 2008 *International Conference onInformation Technology*.

[8] Taranin, S. M. (2016). Backup with Storage in a Database. *Modelirovanie i Analiz Informatsionnykh Sistem*, cilt 23, no. 4, pp. 479-481

[9] Aryan, W., Diana, Subekti, M., & Hendro (2018). Building Scalable and Resilient Database System to Mitigate Disaster and Performance Risks, *Procedia Computer Science*, cilt 135, pp. 25-34

[10] Srinivas, A., Ramayya, Y.S., & Venkatesh, B. (2013). A Study on Cloud Computing Disaster Recovery. *International Journal of Innovative Research in Computer and Communication Engineering*, cilt 1, no. 6, pp. 1380-1388

[11] Bhattacharya, S., Mohan, C., Brannon, K. W., Narang, I., Hsiao, H., & Subramanian, M. (2002). Coordinating Backup/Recovery and Data Consistency Between Database and File Systems, *ACM SIGMOD international conference on Management of data.*

[12] Jin D., & Wang, Q. (2021). CDP Backup and Recovery Method for Ensuring Database Consistency, *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA)*.

[13] Kuyumdzhiev, I. (2019). Comparing Backup and Restore Eficiency in Mysql, *19th SGEM International Multidisciplinary Scientific GeoConference EXPO*.

[14] Sahinaslan, E., Sahinaslan, O., & Bagislanan, O. (2021). Investigation of cost and labor gains achieved by virtualization technologies, *AIP Conference Proceedings*, cilt 2334, no. 070001.

[15] Xia, R., Yin, X., López, Machida, F., & Trivedi, K.S. (2014). Performance and Availability Modeling of ITSystems with Data Backup and Restore. *IEEE Transactions on Dependable and Secure Computing*, cilt 11, no. 4, pp. 375-389

[16] DBCC Ind and DBCC Page. Retrieved January 31, 2016, from https://kwelsql.wordpress.com/2016/01/31/dbcc-ind-and-dbcc-page/, KWEL SQL

[17] DBCC CHECKDB (Transact-SQL) | Microsoft. Retrieved June 8, 2021, from https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-checkdb-transact-sql?view=sql-server-ver15

[18] Kalen , D., Adam, M., Paul R.S., Kimberly, T. L., Conor, C., & Ben, N.(2009). Microsoft SQL Server 2008 Internals, *Microsoft Press.*

[19] Microsoft SQL Server:Restore and Recovery Overview (SQL Server). Retrieved Nov 30, 2021, from https://github.com/MicrosoftDocs/sql-docs/blob/live/docs/relational-databases/backup-restore/restore-and-recovery-overview-sql-server.md.

[20] DELL. *RAID düzeyleri ve teknik özellikleri nelerdir?* https://www.dell.com/support/kbdoc/tr-tr/ 000128635/dell-sunuculari-raid-duzeyleri-ve-teknik-ozellikleri-nelerdir.