TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ DANIŞMANI VE TEZ KONUSU ÖNERME FORMU - EK

1. **TEZ BAŞLIĞI:** Güvenlik Farkındalıklı Veritabanı Göçü Planlaması

2. **KONU VE KAPSAM:** Veritabanı göçü, büyük veri üstüne çalışan firmalar için önemli bir problemdir. Veritabanı göçü hem maliyetli bir işlemdir hem de ciddi güvenlik riskleri içermektedir. Bu nedenle, veri göçü işleminde, bazı firmalar uygulama testlerinden kaynaklanan maliyeti düşürmeyi amaçlarken, göç ettirilecek verinin gizli ve/veya hassas bilgiler içerdiği durumlarda güvenlik risklerini en aza indirmek amaçlanmaktadır.

   Literatürde, depolama türleri, biçimleri veya bilgisayar sistemleri arasındaki veri aktarma işlemi "veri göçü" olarak adlandırılmaktadır [1,2]. Kuruluşlar veya bireyler bilgisayar sistemlerini değiştirdiklerinde veya güncellediklerinde ya da iki firmanın birleşmesi sonucu bilgisayar sistemlerini birleştirmesi gerektiğinde veri göçüne ihtiyaç duyulur. Bu süreçte, veri göçü işlemlerini [3] otomatikleştirdiği [4] ve dolayısıyla da süreçteki insan gereksinimi azalttığı [5] için yazılım çözümleri tercih edilmektedir. Veri göçü işlemi [6,7,8] birden çok aşamadan oluşabilir fakat minimal olarak *veri çıkarımı*, ve *veri yükleme* aşamalarını içerir. Veri göçünün efektif bir şekilde gerçekleştirilebilmesi için verinin eski sistemden yeni sisteme eşleştirilmesi gerekir [9]. Veri çıkarımı ve yükleme prosedürü, be eşleşmeye uygun olacak şekilde belirlenir. Özellikle yeni sistemin güncellemeden ziyade başka bir firmanın yazılımı olduğu durumlarda, bu prosedür karmaşık veri dönüşümleri içerir.

   Depolama göçü [10,11], en sık gerçek gerçekleştirilen veri göçü türüdür ve genellikle firmalar daha verimli depolama teknolojilerine geçiş yapmak istediklerinde gerçekleştirilir. Bunun sonucunda, veri bulunduğu diskten daha efektif bir depolama teknolojisine taşınır. Aynı veritabanı sistemi (ya da dosya sistemi) kullanıldığı için genellikle verinin formatı ve içeriği değişmemektedir. Bu nedenle kullanılan uygulamalar etkilenmez, dolayısıyla da uygulamaların test edilmesi gerekmez.

   Veritabanı göçü operasyonu, firmaların farklı bir veritabanı sağlayıcısına geçiş yapmaları durumunda ya da kullandıkları veritabanı yazılımının güncellenmesi sonucunda da gerçekleştirilir. İlk durumda fiziksel veri göçü işlemine ihtiyaç duyulurken, ikinci durumda genellikle fiziksel veri göçü işlemine ihtiyaç duyulmaz. Veritabanı göçlerinde, göç ettirilen veritabanlarını kullanan uygulamaların nasıl etkileneceği de göz önüne alınmalıdır. Kullanılan veri işlemi dili veya protokolü değiştiğinde, uygulamanın çalışması esnasında istenmeyen davranışlar veya uygulama performansında beklenmeyen değişimler gözlemlenebilir. Bu sebeple, göç ettirilen veritabanlarını kullanan tüm uygulamalar test edilmelidir [12]. Literatürde, veritabanı göçü planlaması problemi test maliyetlerinin en aza indirgenmesi amacıyla incelenmiştir [12]. Bu tezde de test maliyetine ters etkiye sahip olan güvenlik risklerinin en aza indirgenmesine odaklanılacaktır. Veri ne kadar uzun süre erişme açık kalırsa, o kadar büyük bir güvenlik ihlal riski oluşturmaktadır. Dolayısıyla bu tezde ele alınan veritabanı göçü planlaması probleminde güvenlik riski göçün tamamlanması gereken *parti* sayısıyla ilişkilendirilmektedir. Bu ilişkiye bağlı olarak, güvenlik riskinin en aza indirgenmesi için veri göçü en az sayıda parti kullanılarak tamamlanmalıdır.

   Notasyon ve Problem Tanımı
   Tezde ele alınan problem, Güvenlik Farkındalıklı Veritabanı Göçü Planlaması, kısaca GFVGP, problemi olarak tanımlanmıştır. GVFGP probleminde, veritabanları, $\mathcal{B}= \{B_1, B_2, \dots, B_n\}$ veritabanları kümesi ve bu veritabanlarının boyutlarını gösteren $w = [w_1, w_2, \dots, w_n]^T$ vektörü problemin girdileridir. Veritabanı göçü partiler halinde gerçekleştirilmektedir. Veritabanı göçünde veritabanları partilere eşleştirilerek, partideki veritabanları aynı anda taşınmaktadır. Bir partide taşınabilecek toplam veritabanı boyutuna partinin kapasitesi denilmektedir. Her bir partinin kapasitesini gösteren $l = [l_1, l_2, \dots, l_n]^T$ vektörü de problemin bir girdisidir. En kötü durumda, tüm veritabanlarının farklı partilerde taşınması gerekebilir.

Her veritabanı B_i ile ilişkilendirilmiş bir planlama zamanı değişkeni s_i bulunmaktadır ve B_i'nin çözümde hangi partide taşınacağını belirtir. Veritabanları arasında atanabilecekleri partileri kısıtlayabilecek zamansal ilişkiler bulunabilmektedir. Tipik kısıtlara ilişkin şu örnekler verilebilir:

a. Veritabanı B_5 ilk 4 partiden birinde taşınmalıdır. Bu kısıt $s_5 \leq 4$ şeklinde ifade edilir.
b. Veritabanı B_2, veritabanı B_3'ten daha önce taşınmamalıdır. Bu kısıt $s_2 \geq s_3$ şeklinde ifade edilir.

Zamansal ilişkiler de girdinin bir parçası olup *m* elemanlı *C* listesiyle gösterilmektedir. İlk örnekteki gibi tek değişken içeren kısıtlara *kesin* kısıtlar, ikinci örnekteki gibi iki değişken içeren kısıtlaraysa *göreceli* kısıtlar denilmektedir. Modellerde k bir tam sayı olmak üzere $s_i \leq k$ ve $s_i \geq k$ şeklindeki kesin kısıtlara izin verilmektedir. Göreceli kısıtlardaysa $s_2 \geq s_3$ şeklindeki gevşek eşitsizliklere ve $s_2 > s_3$ şeklindeki sıkı eşitsizliklere izin verilmektedir.

Dolayısıyla, GFVGP probleminin girdisi $\langle w, l, C \rangle$ üçlüsünden oluşmaktadır. Aşağıdaki Örnek 1, GFVGP problemi örneği için girdiği göstermektedir.

Örnek 1: $\langle (1,2,2,3)^T, (5,7,10,12)^T, [s_1 < s_2, s_2 \leq s_3, s_4 \leq 3, s_4 \geq 3] \rangle$
Bu örnekte, 1,2,2 ve 3 boyutlarında 4 veritabanı bulunmaktadır. *l* vektörü partilerin kapasitelerini göstermektedir dolayısıyla partilerde taşınabilecek toplam veritabanı boyutları sırasıyla 5,7,10 ve 12'dir. *C* listesi 4 zamansal kısıt içermektedir. İlk kısıt sıkı göreceli zaman kısıtıyken, ikinci kısıt bir gevşek göreceli zaman kısıtıdır. Üçüncü ve dördüncü kısıtlarsa kesin zaman kısıtlarıdır ve beraber veritabanı B_4'ün 3. partide taşınması gerektiğini belirtirler. $S = [s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 3]$ *Örnek 1*'in olası bir çözümüdür. Bu çözümde B_1 veritabanı, 1. partiye, B_2 ve B_3 veritabanları 2. partiye, B_4 veritabanı ise 3. partiye atanmıştır.

Bu problem girdilerin yapılarına bağlı olarak, gerçek hayatta farklı GFVGP problemleri ile karşılaşılmaktadır. GFVGP probleminin temelde dört boyut üzerinde farklılaştığı görülmektedir: (i) veritabanlarının boyutları, (ii) parti kapasiteleri, (iii) veritabanları arasındaki ilişkiler, (iv) amaç fonksiyonu. Bu dört boyutu dikkate alacak şekilde GFVGP model çatısı aşağıdaki gibi tanımlanmıştır.

Model Çatısı

(i) Veritabanlarının Boyutları (**α**) – Bir veritabanının göç etme süresi boyutuyla doğru orantılıdır. Veritabanlarının boyutlarına ilişkin aşağıdaki iki modeli kullanılıyor.

a. Sabit(**sbt**) – Bu modelde, bütün veritabanlarının boyutları eşittir. Bu model, aşağı yukarı aynı boyutta veritabanlarına sahip şirketlerin durumlarını modellemektedir.

b. Keyfi(**kyf**) – Bu modelde farklı veritabanlarının boyutları arasında bir ilişki yoktur. Bu model farklı boyutlarda veritabanlarına sahip şirketlerin durumlarını modellemektedir.

Parti Kapasitesi (**β**) – Veritabanı göçü işlemi esnasında fiziksel olarak taşınan veritabanları erişime kapatılmaktadır. Dolayısıyla bankalar gibi mesai saatlerinde yoğun hizmet veren şirketler, veritabanı göçü operasyonunu hafta sonları veya hafta içi akşam saatlerinde gerçekleştirmeyi tercih ederler. Facebook ve Youtube gibi dünyanın her tarafında ve dolayısıyla farklı zaman dilimlerinde yaşayan kullanıcılara sahip ve haftanın her günü yoğun olarak veritabanlarına erişim olan şirketler için hiçbir zaman dilimi diğer bir zaman dilimine göre veritabanı göçü için daha uygun değildir. Parti kapasitelerine ilişkin aşağıdaki iki model kullanılıyor.

c. Homojen(**hmj**) – Bu modelde bütün parti kapasiteleri eşittir. Bu model Facebook ve Youtube gibi veritabanı erişim oranının homojen dağıldığı şirketlere daha uygundur.

d. Heterojen(**htr**) – Bu modelde parti kapasiteleri arasında bir ilişki yoktur. Bu model veritabanı erişim oranının bankalar gibi heterojen dağıldığı şirketlere daha uygundur.

(ii) Veritabanları arasındaki ilişkiler (**γ**) – Farklı veritabanlarının sakladıkları veriler birbirlerini tamamlayıcı ya da ikame edici nitelikte olabilir. Kullanıcılar tamamlayıcı nitelikteki veritabanlarına eşzamanlı olarak erişim yaptıklarından bu nitelikteki veritabanlarının aynı partide göç etmesi hizmet kalitesi açısından daha uygun olacaktır. Youtube gibi kullanıcıların yüklediği videoları saklayan şirketlerde ise birbirini ikame edici nitelikte içeriklere sahip veritabanları bulunmaktadır. Tipik bir Youtube kullanıcısı, aradığı anahtar kelimelerle alakalı videolar izlemek isteyecektir. Hizmet kalitesi gereği benzer içeriklere sahip farklı veritabanlarının aynı zamanda erişilmez olmaması ve dolayısıyla farklı partilerde göç etmeleri gerekmektedir. Veritabanları arasındaki zamansal kısıtlara ilişkin aşağıdaki üç model kullanılıyor.

a. Yok (**yok**) – Bu model hiçbir zamansal kısıtın olmadığı, her veritabanının herhangi bir partide göç edebildiği durumları modellemektedir

b. Kesin (**ksn**) – Bu modelde sadece $\boldsymbol{s_i \leq 4}$ gibi kesin zamansal kısıtlar bulunabilir.

c. Göreceli (**grc**) – Bu modelde hem kesin hem de göreceli (iki değişkenden oluşan) zamansal kısıtlar bulunabilir. Örnek olarak $\boldsymbol{B_i}$ **ve** $\boldsymbol{B_j}$ veritabanlarının içerikleri birbirini tamamlayıcı nitelikte ise $\boldsymbol{s_i = s_j}$ kısıtı, birbirini ikame edecek nitelikteyse $\boldsymbol{s_i \leq s_j - 1}$ veya $\boldsymbol{s_i \geq s_j + 1}$ göreceli kısıtlarından biri tercih edilebilir.

(iii) Amaç Fonksiyonu(**Θ**) – Şirketlerin farklı önceliklerine ilişkin aşağıdaki iki model kullanıyor.

a. Kullanılan parti sayısı (**kul**) – Kullanılan parti sayısını minimize amaçlayan bu model, veri güvenliği savunmasızlığını (açığı) için önerilmiştir. Veritabanı göçü esnasında, teknik personelin taşınacak tüm veritabanlarına erişimi olacaktır. Teknik personel veritabanı göçü operasyonunu mümkün olan en yüksek güvenlik önlemleriyle gerçekleştirmek için uygun eğitime sahip olsalar bile, gerçekleşen operasyon sonuncunda işlem rutin hale gelecektir. Bu durum gerçekleştiğinde, personelin dikkatsizleşmesi ve veritabanı göçünün güvenlik önemlerini tam anlamıyla uygulamaması potansiyel bir risk haline gelecektir. Hatayla da olsa veri sızmalarının ciddi sonuçları olacağı için veritabanı göçü sık sık tekrarlanmamalıdır. Bu amaç fonksiyonu, potansiyel kötü sonuçları göç sayısını en azlayarak engellemeyi amaçlar. Bu modelde, eğer 1. ve 3. partilere veritabanı atanırsa fakat 2. partiye atanmazsa kullanılan parti sayısı 2 olacaktır.

b. Son kullanılan dizin (**son**) – Bu amaç fonksiyonu göç işlemini en kısa zamanda bitirmek isteyen firmalar için tasarlandı. Bu durum özellikle firmaların bitiş zaman aralıkları için para ödediği durumlarda ortaya çıkmaktadır. Böyle durumlarda firmalar partileri kullansalar da kullanmasalar da bu partiler için ücretlendirilmektedirler. Bu doğrultu da amaç veritabanı göçünü mümkün olduğunca hızlı bir şekilde tamamlamaktır. Bu modelde, eğer 1. ve 3. partilere veritabanı atanırsa fakat 2. partiye atanmazsa kullanılan parti sayısı 3 olacaktır.

Dolayısıyla, bir GFVGP probleminin ait olduğu problem sınıfı $\langle \boldsymbol{\alpha}| \boldsymbol{\beta}| \boldsymbol{\gamma}| \boldsymbol{\theta} \rangle$ dörtlüsü ile gösterilir. Örneğin $\langle \boldsymbol{sbt}|\boldsymbol{hmj}|\boldsymbol{yok}|\boldsymbol{kul} \rangle$, GFVGP probleminin bütün veritabanı boyutlarının eşit, parti kapasitelerinin homojen olduğu, veritabanları arasında zamansal kısıtların bulunmadığı ve amaç fonksiyonun kullanılan son dizini en azlamak olduğu problem sınıfını temsil etmektedir. Önerilen model çatısı, veritabanları boyutları için iki model, parti kapasiteleri için iki model veritabanları arası ilişkiler için üç model ve iki farklı amaç fonksiyonu olduğu için veritabanı göçü planlaması için gerçek hayatta karşılaşılan toplam 2*2*3*2= 24 farklı problem sınıfına cevap vermektedir.

3. **ANAHTAR KELİMELER:** Veritabanı göçü, Algoritma Tasarımı, Sezgisel Algoritmalar, Karma Tam Sayılı Programlama

4. **AMAÇ VE HEDEFLER:**
Tez kapsamında 3 amaç ve bunlara bağlı 3 hedef tanımlanmıştır.

Amaç 1: Önerilen model çatısı altındaki bütün problem sınıflarının hesaplama karmaşıklığının teorik olarak belirlenmesi.

Hedef 1: Problem sınıflarının her birinin hangi hesaplama karmaşıklığı sınıfına dahil olduğunun tespit edilmesi (P ya da NP-zor). P hesaplama karmaşıklığındaki problem sınıfları için polinom zamanlı algoritmaların geliştirilmesi

Amaç 2: NP-zor olduğu gösterilen problem sınıflarının küçük boyutlu problem örneklerini makul sürelerde optimal çözen yöntemlerin geliştirilmesi.

Hedef 2: NP-zor olduğu gösterilen problem sınıflarının küçük boyutlu problem örneklerini çözmeye yönelik etkin optimizasyon modelleri geliştirilecektir.

Amaç 3: NP-zor olduğu gösterilen problem sınıflarının orta ve büyük boyutlu problem örnekleri için sezgisel ve metasezgisellerin geliştirilmesi.

Hedef 3: NP-zor olduğu gösterilen problem sınıflarının orta ve büyük boyutlu GFVGP problem örneklerine makul sürelerde optimalden en fazla %20 uzaklıkta çözümler elde eden sezgisel ve metasezgisellerin geliştirilmesi.

5. **YÖNTEM:** Önerilen çalışma üç ana başlık altında özetlenebilir. Bu başlıklar ve her biri için kullanılacak yöntemler aşağıda sıralanmıştır.

Hesaplama Karmaşıklığı Tespiti: Bu çalışmada ilk çözülmesi gereken sorun hangi problem sınıfları için problemin NP-zor olduğunu anlamak olacaktır. Bunun için kullanılacak başlıca teknik, NP-zorluğu bilinen başka bir problemin Karp [13] indirgemesiyle problemimize dönüştürülebilir olduğunu göstermek olacaktır. Karp indirgemesi yöntemiyle istenilen sonuçlar alınamadığı durumlarda Turing indirgemesi ve doğruluk-tablosu indirgemesi tekniklerini kullanılacaktır.

Küçük Boyutlu Problem Örnekleri için Kesin Çözüm Yöntemlerinin Geliştirilmesi: Her bir problem sınıfı için küçük boyutlu problem örnekleri için optimal çözüm elde etmek amacıyla kesin çözüm yöntemleri geliştirilecektir. Bu amaçla, öncelikle etkin optimizasyon modelleri (hangi veritabanının hangi partide taşınacağını belirleyen ikili (0-1) değişkenler içeren Karma Tamsayılı Programlama (KTP) modelleri) geliştirilecek ve bu modeller küçük problem örnekleri üzerinde geçerlenecek ve geçerli eşitsizlikler (valid inequalities) eklenerek güçlendirilecektir. Herhangi bir problem sınıfında küçük boyutlu problem örnekleri için matematiksel modellerin makul sürede optimal çözüm bulmada yetersiz kalması durumunda ilgili problem sınıfı için kesin çözüm yöntemi olarak Dal-ve-Kesi (Branch-and-Cut), Dal-ve-Ucret (Branch-and-Price) gibi kesin çözüm yöntemlerinin geliştirilmesi planlanmaktadır.

Orta ve Büyük Boyutlu Problem Örnekleri için Sezgisel Çözüm Yöntemlerinin Geliştirilmesi: Kesin çözüm yöntemlerinin orta ve büyük ölçekli problem örnekleri için yetersiz kaldığı NP-zor optimizasyon problemleri için literatürde uygulanan yaklaşım sezgisel ve metasezgisel yöntemler geliştirilmesidir. Yukarıda anlatılan çalışma ile belirlenen her bir problem sınıfı için etkin sezgisel ve/veya metasezgiseller geliştirilmesi planlanmaktadır. Bu kapsamda öncelikle her bir problem sınıfı için problemin en çok benzeştiği optimizasyon problemleri için geliştirilen sezgiseller araştırılacak ve probleme özgü bileşenler dikkate alınarak bir veya birden fazla (karşılaştırma amacıyla) sezgisel veya metasezgisel yöntem geliştirilecektir. Orta ve büyük ölçekli veri setleri üzerinde geliştirilen yöntemlerin hassasiyet analizleri yapılacak ve performansları test edilecektir.

6. REFERANSLAR:

[1] Drumm, Christian, et al. "Quickmig: automatic schema matching for data migration projects." *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 2007.

[2] Gandhi, Rajiv, et al. "Improved results for data migration and open shop scheduling." *ACM Transactions on Algorithms (TALG)* 2.1 (2006): 116-129.

[3] Frigioni, Daniele. *Algorithm Engineering: 5th International Workshop, WAE 2001 Aarhus, Denmark, August 28-31, 2001 Proceedings*. Vol. 5. Springer Science & Business Media, 2001.

[4] Mc Brien, Peter, and Alexandra Poulovassi. "Automatic migration and wrapping of database applications—a schema transformation approach." *International Conference on Conceptual Modeling*. Springer, Berlin, Heidelberg, 1999.

[5] Meier, Andreas. "Providing database migration tools-a practicioner's approach." *Proceedings of the 21th International Conference on Very Large Data Bases*. 1995.

[6] Goldman, Roy, Jason McHugh, and Jennifer Widom. "From semistructured data to XML: Migrating the Lore data model and query language." (1999).

[7] Golubchik, Leana, et al. "Data migration on parallel disks." *European Symposium on Algorithms*. Springer, Berlin, Heidelberg, 2004.

[8] Hall, Joseph, et al. "On algorithms for efficient data migration." *SODA*. Vol. 1. 2001.

[9] Behm, Andreas, Andreas Geppert, and Klaus R. Dittrich. *On the migration of relational schemas and data to object-oriented database systems*. Universität Zürich. Institut für Informatik, 1997.

[10] Hirofuchi, Takahiro, et al. "A live storage migration mechanism over wan for relocatable virtual machine services on clouds." *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE, 2009.

[11] Narayanan, Dushyanth, et al. "Migrating server storage to SSDs: analysis of tradeoffs." *Proceedings of the 4th ACM European conference on Computer systems*. 2009.

[12] Subramani, K., Bugra Caskurlu, and Alvaro Velasquez. "Minimization of Testing Costs in Capacity-Constrained Database Migration." *International Symposium on Algorithmic Aspects of Cloud Computing*. Springer, Cham, 2018.

[13] Karp, Richard M. "Reducibility among combinatorial problems." *Complexity of computer computations*. Springer, Boston, MA, 1972. 85-103.