AZƏRBAYCAN RESPUBLİKASI ELM və TƏHSİL NAZİRLİYİ

AZƏRBAYCAN TEXNİKİ UNİVERSİTETİ

Əlyazması hüququnda

Məmmədov Abdulməcid Güloğlan oğlu

Hava Limanı məlumat bazasının yaradılması

mövzusunda

MAGİSTRLİK DİSSERTASİYASI

İxtisas: 060509 – Kompüter elmləri

İxtisaslaşma: Sistem proqramlaşdırılması

Kafedra müdiri: t.e.d.,prof Ağayev N.B.

Elmi rəhbər: t.e.n. dos. Ağamalıyeva C.A.

BAKI-2023

III FƏSİL. Hava limanı məlumat bazasının yaradılma prosesi

IV FƏSİL. Python proqramlaşdırma dilində yaratdıqım hava limanı məlumat bazsının kod hissəsi.

Giriş

I FƏSİL. Aerportun məlumat bazasının yaradılmasının məqsədləri.

1.1. Məlumatların yığılması:

Aerportun məlumat bazasında məlumatların yığılması, hava limanında tələb olunan bütün məlumatların toplanması və saxlanılması işidir. Bu, hava limanının fəaliyyətinin daha effektiv idarə edilməsi və iş proseslərinin daha yaxşı nəzarət edilməsi üçün əhəmiyyətli bir addımdır.

Məlumatların yığılması, hava limanında işləyən müxtəlif sistemlərdən, cihazlardan və sənədlərdən cəmləşdirilə bilər. Bu, pasajların, yüklərin və uçuşların informasiyasını, hava limanının əhalisi ilə bağlı məlumatları, maliyyə informasiyasını və digər bir çox məlumatı daxil edir. Bu məlumatlar elektronik sistemlərdə, sənədlərdə və cihazlarda cəmləşdirilə bilər.

Məlumatların yığılması prosesi zamanı, məlumatlar ayrılmaz bir şəkildə saxlanılmalı və təhlil edilməlidir. Bu, məlumatların doğruluğunun təmin edilməsinə və onların effektiv şəkildə istifadə edilməsinə kömək edir. Məlumatların yığılması zamanı, məlumatlar həmçinin avtomatik olaraq tənzimlənməlidir və tələb olunan formata uyğun olaraq qruplandırılmalıdır.

Aerportun məlumat bazasının yaradılması zamanı, məlumatların gizliliyi və təhlükəsizliyi son dərəcə əhəmiyyətli məsələlərdir. Məlumatların qorunması, hava limanında əməliyyatların əsasında yer alan məlumatların müstəqil bir şəkildə saxlanılması və hava limanının ümumi təhlükəsizliyinin qorunması üçün əsas rol oynayır.

Məlumatların gizliliyi, məlumatların yalnız təyin edilmiş şəxslər tərəfindən girişinin təmin edilməsi ilə təmin edilir. Bu məqsədlə, məlumat bazası təhlükəsizlik tədbirləri ilə müdafiə olunmalıdır, həmçinin təyin edilmiş istifadəçilərin təyin olunan səviyyədə istifadəsi ilə nəzarət olunmalıdır. Bu, gizlilik qaydalarına uyğun bir şəkildə informasiyanın saxlanılmasını təmin edir.

Məlumatların təhlükəsizliyi isə informasiyanın müxtəlif formalarının üzərindəki mənsubiyyəti və informasiyanın gizlilik qaydalarına uyğun istifadə edilməsi ilə bağlıdır. Məlumat bazasının təhlükəsizliyini təmin etmək üçün, məlumatlar müvafiq təhlükəsizlik tədbirləri ilə müdafiə olunmalıdır. Bu, informasiyanın qəsdən yaxasına çıxarılmamasına, şəxsiyyət hırsızlığına, və ya hər hansı bir təhlükəsizlik açığına yol açacaq hər hansı bir aktivlikdən qoruyacaqdır.

Ayrıca, məlumat bazasının təhlükəsizliyi və gizliliyi üçün, informasiyanın dəyişdirilməsinə və ya silinməsinə qarşı müdafiə mexanizmləri də təyin edilməlidir. Bu mexanizmlər, informasiyanın üzərindəki mənsubiyyətinə dair qaydaların pozulması, müdaxilə, qəsdən xərclənmə və ya informasiyanın təsəvvürsüz silinməsi kimi təhlükəsizlik açıqlarına qarşı tədbirlər təmin edir.

1.2. Məlumatların tənzimlənməsi:

Aerportun məlumat bazasında məlumatların tənzimlənməsi, məlumatların standart bir şəkildə təqdim edilməsini və saxlanılmasını təmin edir. Bu, məlumatların daha effektiv istifadə edilməsinə imkan verir və hava limanının idarəetməsini asanlaşdırır.

Məlumatların tənzimlənməsi, məlumatların bəzi əsas kateqoriyalarının yaradılması ilə başlayır. Bu kateqoriyalar, məlumatların uyğun bir şəkildə indekslənməsinə və təqdim edilməsinə imkan verir. Örnək olaraq, hava limanında səyahət edən sərnişinlərlə bağlı məlumatlar, uçuşlarla bağlı

məlumatlar, sərnişin baqajları ilə bağlı məlumatlar kimi fərqli kateqoriyalara ayrıla bilər. Hər bir kateqoriya, məlumatların daha da spesifik və müvafiq indekslənməsinə imkan verən alt-kateqoriyalara ayrıla bilər. Bu, məlumatların daha effektiv şəkildə indekslənməsinə imkan verir və hava limanının digər bölgələri, müştəriləri və ya digər təşkilatlarla informasiyanın mübadiləsini asanlaşdırır. Məlumatların tənzimlənməsi, həm daxili, həm də xarici informasiya mübadilələri üçün standart formaların təyin edilməsini də əhatə edir. Bu formalar, məlumatların müvafiq və standart bir şəkildə təqdim edilməsini təmin edir və hava limanının digər bölgələri, müştəriləri və ya digər təşkilatlarla informasiyanın mübadiləsini daha asanlaşdırır. Nəticə olaraq, məlumatların tənzimlənməsi, aerportun məlumat bazasının effektivliyinin artırılması üçün əhəmiyyətli bir amildir. Bu, məlumatların standart bir şəkildə indekslənməsi və təqdim edilməsi vasitəsilə hava limanının ümumi idarəetməsini asanlaşdırır və təcrübələri optimallaşdırır.

1.3. Məlumatların təqdim edilməsi:

Aerportun məlumat bazası, müştərilərə və hava limanı işçilərinə tələb olunan bütün məlumatların asanlıqla təqdim edilməsinə imkan verəcək. Bu, hava limanının effektiv idarə olunması və müştərilərə daha yaxşı xidmət göstərilməsi üçün əhəmiyyətli bir addımdır.

1.4. İdarəetmənin effektivləşdirilməsi:

Aerportun məlumat bazası, hava limanının idarəetmə proseslərinin daha effektiv və sürətli şəkildə idarə olunmasına kömək edəcəkdir. Bu, hava limanının effektivliyini artıracaq və işlərində sürətli və təsirli bir şəkildə nəzarət edilməsinə imkan verəcək. Aerportun məlumat bazasının yaradılması, hava limanı idarəetməsinin effektivliyinin artırılmasına yardım edir. Məlumat bazasında

saxlanılan məlumatların indekslənməsi və tənzimlənməsi, idarəetmənin daha sürətli və effektiv olmasını təmin edir. Bu, idarəetmə proseslərinin daha da avtomatikləşdirilməsini, səhv risklərinin azaldılmasını və müştərilərin xidmət səviyyəsinin artırılmasını mümkün edir.

2. Aerportun məlumat bazasının idarəetməni effektivləşdirmək üçün bir neçə yol var:

2.1. Məlumatların real vaxtında saxlanılması: Məlumat bazasında saxlanılan məlumatların real vaxtında yenilənməsi, idarəçilərə vaxtında məlumatlar verir və hər hansı bir səhv riskini azaltır. Aerportun məlumat bazasında məlumatların real vaxtında saxlanılması, hava limanının idarəetməsinin effektivliyini artırır. Real vaxtda məlumatların yenilənməsi, idarəetmənin daha sürətli və riskləri az olan bir şəkildə həyata keçirilməsini təmin edir. Aerportun məlumat bazasında saxlanılan məlumatların real vaxtında yenilənməsi, məlumatların təhlükəsizliyini də təmin edir. Məlumatlar güclü şifrələmə sistemləri ilə qorunur və yalnız səlahiyyətli şəxslər tərəfindən giriş edilə bilər.

Real vaxtda məlumatların yenilənməsi, hava limanının müştərilərinə daha sürətli və dəqiqlik ilə xidmət göstərməsinə kömək edir. Misal üçün, uçuşların statusunun real vaxtda yenilənməsi, yolçulara daha dəqiqlik ilə məlumat verir və uçuşla bağlı hər hansı bir dəyişiklikdən vaxtında xəbərdar edir. Real vaxtda məlumatların yenilənməsi, hava limanı əməkdaşlarına daha sürətli və effektiv bir şəkildə işləmək imkanı verir. Misal üçün, uçuş məlumatlarının real vaxtda yenilənməsi, uçuşların idarə edilməsinin daha sürətli və səhv riskləri az olan bir şəkildə həyata keçirilməsinə kömək edir. Bütün bunlar, Aerportun məlumat bazasında məlumatların real vaxtında saxlanılmasının, hava limanının idarəetmə proseslərində daha sürətli, effektiv və dəqiqlik ilə həyata keçirilməsinə kömək etdiyini göstərir.

2.2. İdarəetmə təlimatlarının avtomatikləşdirilməsi: Məlumat bazasında saxlanılan məlumatlar, avtomatik idarəetmə təlimatlarının yaradılmasını mümkün edir. Aerportun məlumat bazasında idarəetmə təlimatlarının avtomatikləşdirilməsi, hava limanının idarəçilik proseslərində effektivliyi artırır və insan səhvlərinin azalmasına kömək edir. Bu avtomatlaşdırma prosesi, hava limanı əməkdaşlarının lazımi təlimatları daha sürətli və səhv riski minimum olan bir şəkildə almasını təmin edir. Bu, idarəetmə proseslərinin daha sürətli və effektiv olmasına kömək edir. İdarəetmə təlimatlarının avtomatikləşdirilməsi, hava limanının işləyiciləri arasında məlumatların daha sürətli və tələbə uyğun şəkildə paylaşılmasına da kömək edir. Bu, işləyicilərin daha sürətli bir şəkildə hər hansı bir problemi həll etməsinə və ya müştərilərə dəqiqlik ilə məlumat verilməsinə imkan verir. Ayrıca, idarəetmə təlimatlarının avtomatikləşdirilməsi, insan səhvlərinin azalmasına da kömək edir. Bu, xüsusilə, işləyicilərin işlərinin bir hissəsini avtomatlaşdırmaqla və hər hansı bir məlumatı əl ilə daxil etmək yerinə, sistemlərin avtomatik olaraq informasiyanı təsdiq etməsinə imkan verir. Bütün bunlar, aerportun məlumat bazasında idarəetmə təlimatlarının avtomatikləşdirilməsinin, hava limanının idarəetmə proseslərində daha sürətli, effektiv və səhv riski minimum olaraq həyata keçirilməsinə kömək etdiyini göstərir.

2.3. Proseslərin təkmilləşdirilməsi: Məlumat bazasında saxlanılan məlumatların analizi, idarəetmə proseslərinin təkmilləşdirilməsinə kömək edir.

Aşağıdakı addımlar proseslərin təkmilləşdirilməsinə kömək edir:

***1. Prosedurları təkmilləşdirin*:** Aerport məlumat bazasının prosesləri mümkün qədər səmərəli və sürətli olmalıdır. Buna görə də, prosedurların yaxşı təşkil edilməsi və proseslərin sürətli həyata keçirilməsi üçün təkmilləşdirmələr edilməlidir

2. Yeniliklərə açıq olun: Müasir texnologiyaların hərəkətə keçirilməsi, proseslərin təkmilləşdirilməsində kömək edir. İnternet, bulud hesablama və digər avtomatlaşdırma texnologiyaları, aerport məlumat bazasının daha səmərəli və sürətli olmasına imkan verir.

3. Səhvləri aradan qaldırın: Aerport məlumat bazasında səhv riskini azaltmaq üçün prosedurları və sistemləri sıxılaşdırın. Bu, səhvlər və təkrarlanan problemləri aradan qaldırmağa kömək edir.

4. Müştəri münasibətlərini qayğılandırın: Aerport məlumat bazası, müştərilərin ənənəvi sorğularına və tələblərinə cavab verə bilməlidir. Bu məqsədlə, müştərilərin tələblərinə uyğun olaraq məlumat bazasında əlavələr və yeniliklər edilməlidir.

5. Proseslərin monitorinqi: Aerport məlumat bazasının proseslərinin monitorinqi, səhv riskini azaltmağa kömək edir. Prosedurların və sistemlərin səmərəliyi, təhlükəsizliyi və effektivliyi düzgün bir şəkildə monitorinq olunmalıdır.

Bu addımlar, aerport məlumat bazasının proseslərinin təkmilləşdirilməsinə kömək edir və hava limanının daha səmərəli və effektiv olmasına imkan verir.

4. Proaktiv həllər: Məlumat bazası, idarəçilərə proaktiv həllər təklif edir. Məlumat bazasında saxlanılan məlumatların analizi, hər hansı bir problemin daha tez bir şəkildə həll edilməsini təmin edir.

Aerport məlumat bazası proaktiv həllər dəstəkləyir, yəni problem yarandıqdan sonra onu həll etmək yerinə, potensial problemləri qabaqcıl şəkildə müdafiə etməyə çalışır. Bu, məlumat bazasının öyrənmə və təhlil funksiyalarını istifadə edərək, uçuşlar, yolcular və digər məlumatlar üzərində ciddi analizlər aparmağı və nəticələrə uyğun olaraq aksiyalar tətbiq etməyi daxil edir.

Məsələn, proaktiv bir həllər prosesi, uçuş planlama və təminatında keyfiyyətin artırılması üçün məlumat bazasındakı istatistiklərə vətəndaşların səyahət məqsədləri, uçuş tarixləri və zamanlamaları kimi faktorları daxil edərək hazırlanır. Buna görə də uçuşlar daha sıxılaşdırılaraq, daha sürətli və səmərəli təmin edilə bilir.

Bu proaktiv həllər ayrıca, zərərli proqramlar və viruslarla mübarizədə də istifadə edilə bilər. Məlumat bazasındakı keyfiyyətli təhlil və xəbərdarlıq sistemi sayəsində zərərli proqramların həyata keçirilməsi qabaqcıl şəkildə mümkündür və birbaşa təsir edən problemləri təhlil edərək, potensial təhlükələri vaxtından əvvəl tanımaq və həll etmək mümkündür.

5. Digər təşkilatlarla inteqrasiya: Aerportun məlumat bazası, digər hava limanları və ya təşkilatlarla informasiya mübadiləsi üçün inteqrasiya edilə bilər. Bu, müştərilərin xidmət səviyyəsinin artırılmasını və idarəetmənin daha effektiv həyata keçirilməsini təmin edir. Aerport məlumat bazası, digər təşkilatlarla inteqrasiya üzrə də dəstək verir. Hava yolu şirkətləri, gömrük şöbələri, hava trafik idarələri, qatarlaşdırma şirkətləri və s. kimi fərqli təşkilatlarla əməkdaşlıq edərək, məlumatların təhlili və paylaşılması üzrə birbaşa kommunikasiya imkanı verir.

Bu inteqrasiya, bir neçə fərqli sistemlər arasında məlumatlarınızın xüsusiyyətlərini bir-birinə paylaşmağı asanlaşdırır. Məsələn, hava yolu şirkəti Aerport məlumat bazasına yolcu səyahət məlumatlarını göndərə bilər və bunlar Aerport məlumat bazasında saxlanılır. Aerport məlumat bazası, daha sonra bu məlumatları hava trafik idarəsi ilə paylaşa bilər və daha sürətli, effektiv uçuş planlama və nəzarət imkanı verir. İnteqrasiya, təşkilatların bir-biriləri ilə fərqli məlumat sistemlərində saxlanılan məlumatları əldə etməyə çalışmadan ən yaxşı

məlumatları əldə etməyə imkan verir. Bu da əlavə təhlil və idarəetmə imkanlarına imkan verir və iş proseslərinin təkmilləşdirilməsinə kömək edir.

Bütün bunlar, Aerportun məlumat bazasının effektivliyini artırır.

5. Qərar vermə proseslərinin təkmilləşdirilməsi:

Aerportun məlumat bazası, hava limanının qərar vermə proseslərinin daha yaxşı şəkildə təkmilləşdirilməsinə kömək edəcəkdir. Bu, hava limanının işlərində daha yaxşı qərarlar verilməsinə imkan verəcək və daha effektiv bir idarəetmə prosesi yaradacaqdır.

Aerport məlumat bazası, qərar vermə proseslərinin təkmilləşdirilməsində də bir çox imkanlar təqdim edir. Məlumatlarının doğru və zamanında təhlili sayəsində, Aerport idarəçiləri məlumat bazası vasitəsi ilə müxtəlif sahələrdə qərar vermək üçün məlumat əldə edə bilərlər.

Məsələn, Aerport məlumat bazasında qeydiyyatdan keçirilmiş olan uçuşlarla bağlı olan məlumatlar, idarəçilərə uçuşların mövcud halını və planını bilmək imkanı verir. Bu məlumatlar, uçuşların gecikməsi, ləğv edilməsi və ya başqa dəyişikliklərə səbəb ola biləcək faktorları müəyyənləşdirmək üçün istifadə oluna bilər. Aerport məlumat bazası, ayrıca fərqli məlumatlar arasında əlaqələr yaratmağı da asanlaşdırır. Bu məlumatlar arasında uçuşlar, bagajlar, yolcular və hətta hava şəraitindən məlumatlar da daxildir. Bu məlumatların hamısı bir arada işlənərək, idarəçilərə daha yaxşı qərarlar vermək üçün tam bir baxış imkanı verir.

Ümumiyyətlə, Aerport məlumat bazası, idarəçilərə daha sürətli və effektiv qərar vermək üçün məlumatları təhlil etmək, birləşdirmək və tənzimləmək üçün bir çox imkanlar təqdim edir. Bu da idarəetmənin effektivliyini artırır və uçuşların vaxtında, səliqəli və təhlükəsiz olaraq həyata keçirilməsinə kömək edir.

II FƏSİL. Məlumat bazasının tədqiqi.

Məlumat bazasının tədqiqi: Bu hissədə, hava limanı məlumat bazasının necə işlədiyinə, hansısa məlumat bazasının funksiyalarına və əsas məlumatların necə qruplandığına diqqət yetirilməlidir.

1. Hava limanı məlumat bazası necə işləyir?

Hava limanı məlumat bazası, hava limanı əməkdaşları tərəfindən istifadə edilən bir informasiya sistemi olaraq işləyir. Bu məlumat bazası, hava limanının fəaliyyətinə dair məlumatların toplanması, saxlanılması, tənzimlənməsi və təhlil edilməsini özündə cəmləşdirir. Hava limanı məlumat bazasında, uçuşlar, sərnişinlər, yüklər, hava şəraitləri, aviasiya təhlükəsizliyi, məhsul-layihələr və digər bir çox məlumatların saxlanılması mümkündür. Bu məlumatlar, hava limanının idarəetmə və inzibati orqanları tərəfindən istifadə olunur. Məlumat bazasının əsas funksiyaları arasında, məlumatların avtomatik yığılması, idarə olunması və işlənməsi, məlumatların qeydiyyatı və saxlanılması, məlumatların təhlili və hesablanması və digər proseslər yer alır.

Hava limanı məlumat bazası, hava limanı əməkdaşları üçün əhəmiyyətli bir araçdır, çünki bu sistem vasitəsi ilə əməkdaşlar hər hansı bir məlumatı asanlıqla tapa bilərlər və bu məlumatları dəqiqliklə işləyə bilərlər. Bu da hava limanının effektivliyini artırır və xərclərin azaldılmasına kömək edir.

Aerport məlumat bazasında hansı məlumat bazasının funksiyalarına diqqət yeirilməlidir?

Aerport məlumat bazasında aşağıdakı məlumat bazası funksiyalarına diqqət yetirilməlidir:

1.1. Məlumatların yığılması və saxlanılması

Aerport məlumat bazasında məlumatların yığılması və saxlanılması, hava limanının bütün fəaliyyətləri ilə əlaqədar məlumatların toplanması, cəmlənməsi və asanlıqla daxil olunacağı bir vəziyyətdə saxlanılması ilə bağlıdır. Bu, hava limanının fəaliyyətinin hər bir hissəsindən məlumatların, hər hansısa bir olay, hadisə və ya fəaliyyət dairəsindən edinilən məlumatların, sərnişinlər və uçuş əməkdaşları ilə əlaqəli məlumatların və digər hər hansısa məlumatın özündə cəmləşdirilməsini əhatə edir. Məlumatların saxlanılması zamanı, həm fiziki, həm də məntiqi təhlükəsizlik önəmlərinə diqqət edilməlidir. Məlumatların saxlanılması üçün hər hansısa bir məlumat bazası proqramı istifadə olunmalıdır. Bu proqram, məlumatların yaddaşda saxlanılması və təhlükəsizliyi ilə əlaqədar olan bütün prosesləri idarə edir və saxlama məsafəsinin azaldılması və məlumatların daha sürətli çatdırılması üçün optomallaşdırma proseslərini həyata keçirir.

1.2. Məlumatların tənzimlənməsi və qruplaşdırılması

Aerport məlumat bazasında məlumatların tənzimlənməsi və qruplaşdırılması, məlumatların effektiv və sürətli axtarışının təmin edilməsi üçün əhəmiyyətli funksiyalardandır. Bunun üçün məlumatların doğru və müvafiq şəkildə təyin edilmiş struktur və kateqoriyalara uyğun olaraq qruplaşdırılması və qaydalar təyin edilməsi lazımdır. Məlumatların qruplaşdırılması sayəsində, həm idarəçilər, həm də digər məlumat bazası istifadəçiləri tərəfindən axtarış zamanı daha rahat və sürətli tapıla bilər. Bu prosesin düzgün həyata keçirilməsi, məlumatların səmərəli idarə olunmasını təmin edir və həm də qrafiklər, statistikalar kimi məlumatların təqdimatını asanlaşdırır.

1.3. Məlumatların axtarışı və filtrasiyası

Aerport məlumat bazasında məlumatların effektiv axtarışı və filtrasiyası çox vacib bir funksiya hesab olunur. Bu, məlumat bazasındakı məlumatların daha sürətli və effektiv bir şəkildə tapılmasına və istifadə edilməsinə imkan verir. Məsələn, bir istifadəçi hava limanında hansı uçuşların mövcud olduğunu və ya hansı istiqamətlərə uçuşların təyin edildiyini axtarır. Axtarış nəticələri filtrlənə və ya qruplaşdırıla bilər, məsələn, qiymətə görə, uçuşun müddətinə görə və ya s.

Ayrıca, məlumat bazasında indeksləmə və optimallaşdırma üsulları istifadə oluna bilər, bu da axtarış və filtrasiya proseslərini daha sürətli və effektiv həlledə bilər. İstifadəçilər axtarış prosesində əlavə filtrlər və ya spesifik məlumat tələbləri daxil edə bilərlər, bu isə təxminən axtarış prosesinin təkmilləşdirilməsi üçün yararlıdır.

1.4. Məlumatların təhlil edilməsi və hesablanması

Aerport məlumat bazasında məlumatların təhlil edilməsi və hesablanması, məlumatların daha dəqiq və ətraflı şəkildə qiymətləndirilməsinə kömək edir. Buna görə, məlumatların statistikası, rəyləri və digər məlumatların hesablanması, həmçinin məlumatların digər təhlili funksiyaları əlavə olunmalıdır. Bu funksiyaların tətbiqi, hava limanı idarəetməsinə, yaradılan məlumat bazası vasitəsilə hava limanının bir çox sahələrində gözlənilən problemlər barədə daha dəqiq məlumatlar verməyə imkan verir.

1.5. Proqram təminatının tənzimlənməsi və yenilənməsi

Aerport məlumat bazasında proqram təminatının tənzimlənməsi və yenilənməsi son dərəcə önəmlidir. Bu, məlumat bazasının əməliyyatlarının düzgün şəkildə həyata keçirilməsi və məlumatların təhlili və idarə edilməsi üçün tələb olunan funksionallığın əldə edilməsi üçün vacibdir.

Proqram təminatının tənzimlənməsi, məlumat bazasının yeniliklərə və yeniləmələrə uyğun hala gətirilməsini təmin edir. Bu, məlumat bazasının performansının, təhlükəsizliyinin və funksionallığının optimal səviyyədə qalmasını təmin edir. Proqram təminatının yenilənməsi, tətbiqlərin müvafiq şəkildə işləməsini və müasir təhlükəsizlik tədbirlərini əldə etməyi təmin edir.

Ayrıca, proqram təminatının tənzimlənməsi və yenilənməsi, yeni funksionallığın və məlumat bazası əməliyyatlarının tətbiq edilməsi ilə bağlı təlimatların yenilənməsinə imkan verir. Bu, məlumat bazası işlərini təkmilləşdirərək effektivliyi artırır və idarəetməni daha sürətli və asan edir.

1.6. Məlumatların təhlükəsizliyi və gizliliyi

Aerport məlumat bazasında məlumatların təhlükəsizliyi və gizliliyi çox önəmlidir. Məlumat bazasında saxlanılan məlumatların təhlükəsizliyi üçün müxtəlif tədbirlər götürülməlidir. Bu tədbirlər arasında məlumat bazasına yetkisiz girişi qarşısında qorunma, verilənlərin qorunması və eyni zamanda baza sisteminin fəaliyyətindən məlumatların itirilməməsi və ya silinməməsi üçün sərbəst yedəkləmələr etmək daxildir. Məlumatların gizliliyi isə baza sistemində saxlanılan məlumatların qarşı tərəflər tərəfindən yetkisiz olaraq əldə edilməsinin qarşısını almaq üçün göstərilən tədbirləri əhatə edir. Bu tədbirlər arasında məlumatların şifrələnməsi, təhlükəsizlik protokollarının tətbiqi və məlumatlara yalnız yetkilər tərəfindən giriş imkanı verilməsi sayılabilir.

Məlumat bazasında məlumatların təhlükəsizliyi və gizliliyi, müştərilər, təşkilatlar və digər tərəflər üçün əhəmiyyətli olduğundan, baza sistemi idarəetməsi bu sahədə müvafiq tədbirləri həyata keçirməlidir.

1.7. Real vaxtında məlumatların saxlanılması

Aerport məlumat bazasında real vaxtında məlumatların saxlanılması, informasiya tələb və təkliflərinə sürətli cavab vermək üçün çox vacibdir. Bu məqsədlə, məlumat bazasında səs-küy, rəng, müəyyən sözlər və ya hər hansısa məlumatın istənilən dərəcədə cəlb edilməsi və saxlanılması üçün sensor və avtomatik sistemlər quraşdırılır. Bu sayədə, hava limanı əməkdaşları və digər təşkilatlar real vaxtda ən son məlumatlardan xəbərdar olmaq imkanına malik olurlar. Həmçinin, real vaxt məlumatları analiz edilərək səyahət prosesi və sərnişinlərin təcrübəsi üçün təkliflər və göstəricilər hazırlanır.

1.8. İş proseslərinin təkmilləşdirilməsi və effektivləşdirilməsi

Aerport məlumat bazasında iş proseslərinin təkmilləşdirilməsi və effektivləşdirilməsi, təşkilatın keyfiyyətli və effektiv olaraq işləməsinə kömək edir. Buna görə, aşağıdakı addımlar atılmalıdır:

1. İş proseslərinin təhlili: Mövcud iş prosesləri və proseslərdən keçirilən məlumatlar analiz edilməlidir. Bu, proseslərdən keçirilən məlumatların axını və bu məlumatların necə işləndiyini təyin etmək üçün vacibdir.

2. İş proseslərinin yenidən dizaynı: Analiz zamanı identifikasiya edilən problemlər üzərində çalışılır və mümkün olan ən yaxşı həllər axtarılır. Yenidən dizayn prosesi, müəyyən prosesləri və funksiyaları təkmilləşdirir, bu da səhmdarların və müştərilərin qarşısında daha effektiv bir təşkilat görünməsinə kömək edir.

3. Proseslərin avtomatlaşdırılması: İş proseslərinin bir çox hissələri avtomatlaşdırıla bilər. Bu, həm keyfiyyəti yüksəltərək səhmdarların birbaşa vaxtının azalması, həm də xərclərin azalması deməkdir.

4. Proseslərin izlənməsi və qiymətləndirilməsi: Prosedurun tətbiq edilməsindən sonra, nəticələrin izlənməsi vacibdir. Həmçinin, proseslərin müəyyən edilmiş

keyfiyyət standartlarına uyğun olub-olmadığı və effektiv olduğu müəyyən edilməlidir. Bu, proseslərin gələcək üçün daha yaxşı dizayn edilməsinə və effektivliyin artırılmasına kömək edir.

2. Aerport məlumat bazasında hansı məlumatların qruplandığına diqqət yetirilməlidir?

Aerport məlumat bazasında müxtəlif məlumatlar qruplandırıla bilər. Bəzi əsas qruplar aşağıdakılardır:

2.1. Sərnişin məlumatları: Sərnişinlərə aid məlumatlar, məsələn adı, soyadı, doğum tarixi, pasportun nömrəsi və səyahət məlumatları bu qrupa daxil edilir. Hava limanına gələn və uçan sərnişinlər haqqında məlumatları, onların səyahət tarixçəsini, bərpa edilmiş bağajlarını, qeydiyyat məlumatlarını və s. daxil edir. Bu məlumatlar, sərnişinlərin səyahət etməsi üçün tələb olunan sənədlər, bilet məlumatları və onların uçuş haqqında başqa məlumatları da əlavə edə bilər. Bu məlumatlar, sərnişinlər üçün daha yaxşı xidmət göstərmək və həyat təhlükəsizliyini təmin etmək üçün önəmlidir.

2.2. Uçuş məlumatları: Bu qrupda uçuşlarla bağlı məlumatlar, məsələn uçuş nömrəsi, uçuşun tarixi və zamani, çatışma və gecikmələr kimi məlumatlar yer alır.

Aerport məlumat bazasında uçuş məlumatları, hər hansısa bir uçuşun təyinat və çıxış nöqtələri, uçuşun vaxtı, uçuşun növü, uçuş təyinatı, uçuş planı, uçuş sənədləri, uçuşla bağlı məlumatlar, uçuşun statusu kimi məlumatları əhatə edir. Bu məlumatlar, sərnişinlərin uçuş haqqında daha ətraflı məlumat almasına, uçuşun idarə olunmasına və təhlükəsizliyin təmin edilməsinə kömək edir. Məlumat bazasındakı uçuş məlumatları, hər hansısa bir uçuşun idarə olunmasına,

proseslərin təkmilləşdirilməsinə və sərnişin tələblərinə cavab verilməsinə kömək edir.

2.3. Bilet məlumatları: Bilet satışı və ödənişi ilə bağlı məlumatlar bu qrupa daxil edilir. Aerport məlumat bazasında bilet məlumatları, sərnişinlərin uçuşlara olan bilet müraciətlərinə və onların bilet məlumatlarına dair məlumatları əhatə edir. Bu məlumatlar, sərnişin adı, bilet nömrəsi, bilet tarixi və qiyməti kimi məlumatları əhatə edir. Bu məlumatlar, hava yolu şirkətləri və sərnişinlər üçün əhəmiyyətli olaraq saxlanılmalıdır və bilet məlumatlarının doğruluğu və tamamlığı, müəyyən standartlar və qaydalar tərəfindən təmin edilməlidir. Həmçinin, bilet məlumatları, müştərilərə dəstək xidmətləri tərəfindən də istifadə edilir və bu məlumatların səhv olmaması və ya itilməməsi əhəmiyyətli olarq qarşılanmalıdır.

2.4. Yük məlumatları: Yük uçuşları ilə bağlı məlumatlar bu qrupa daxil edilir.

Aerport məlumat bazasında yük məlumatları, hava yolu şirkətlərinin yüklərini hərəkət etdirməklə bağlı məlumatlar və hava limanı lojistik proseslərinin idarə edilməsi üçün lazım olan məlumatları əhatə edir. Bu məlumatlar, yüklərin hansı terminallarda və hansı uçuşlarda yerləşdiriləcəyi, hər yükün ölçüsü və növü, yükün hansı hava yolu şirkəti tərəfindən göndərildiyi və qəbul edildiyi, həmçinin yüklərin hansı vasitələrlə hansı istiqamətlərə çatdırılacağı kimi məlumatları daxil edir. Bu məlumatlar hava limanı və hava yolu şirkətlərinin lojistik proseslərini daha effektiv edir və yüklərin sürətli və effektiv bir şəkildə hərəkət etməsini təmin edir.

2.5. Texniki məlumatlar: Hava vasitələri ilə bağlı məlumatlar, məsələn hava vasitəsinin nömrəsi, modeli və texniki xüsusiyyətləri bu qrupa daxil edilir.

Aerport məlumat bazasında texniki məlumatlar hava limanında istifadə edilən aviaşirkətlərə, qatarlar və digər hava vasitələrinə dair məlumatları əhatə edir. Bu məlumatlar, hava limanı işçiləri və ya texniki personel tərəfindən istifadə olunur və aviaşirkətlərin hava vasitələrinin baxım və tamir proseslərini idarə etməkdə köməklik göstərir. Məlumat bazasında ən çox qeyd olunan texniki məlumatlar arasında hava vasitələrinin idarəetmə sistemləri, maşınlarının baxım planları, müvəqqəti və ya uzunmüddətli qadağalar, aviaşirkətlərin hava vasitələri üçün tələb olunan sənədlər və s. olur. Bu məlumatlar, hava limanı işlərində sazlamalar, planlaşdırmalar və bütün texniki proseslər üçün əhəmiyyətli ola bilər.

2.6. İdarəetmə məlumatları: İdarəetmə prosesləri ilə bağlı məlumatlar, məsələn personalların işləyisi, məhsulların depo dərəcəsi kimi məlumatlar bu qrupa daxil edilir. Aerport məlumat bazasında idarəetmə məlumatları, hava limanı işləri üzərində nəzarət və idarəçilik funksiyalarının icrası üçün dəyərli məlumatlar təqdim edir. Bu məlumatlar arasında hava limanı təhlükəsizliyi ilə bağlı məlumatlar, zəruri ehtiyacların sifarişi və təchizatının idarə olunması, personel təlimləri, işgüzar effektivliyi, maliyyə hesabatları, idarəetmə planları və s. yer alır. İdarəetmə məlumatları, hava limanının işlərini daha effektiv və sürətli həyata keçirməyə kömək edir və müştərilərin, təşkilatların və qurumların gözləntilərini qarşılamaya imkan verir.

Bu məlumatlar və daha bir çoxu, Aerport məlumat bazasında uyğun qruplara daxil edilir və burada saxlanılır. Bu, məlumatların düzgün tənzimlənməsinə və effektiv idarə edilməsinə imkan verir.

III FƏSİL. Hava limanı məlumat bazasının yaradılma prosesi.

Hava limanı məlumat bazasının yaradılma prosesi, birdən çox mərhələdən ibarətdir. Bu mərhələlər aşağıdakılar olmaqla yanaşı, əlavə mərhələlər də ola bilər:

1. Tələblərin təyini: Hava limanı məlumat bazasının yaradılması prosesi bir şirkətin tələblərinə uyğun olmalıdır. Bu mərhələdə, hava limanının nələrə ehtiyacı olduğu müəyyənləşdirilir və məlumat bazasının nə üçün istifadə olunacağına qərar verilir.

2. Məlumatların təsis edilməsi: Hava limanı məlumat bazası üçün lazım olan məlumatlar müəyyənləşdirilir. Bu məlumatlar, sərnişin məlumatları, uçuş məlumatları, bilet məlumatları, yük məlumatları, texniki məlumatlar, idarəetmə məlumatları kimi hava limanı ilə əlaqədar olan bütün məlumatlar ola bilər.

1. Aerport məlumat bazası üçün məlumatların təsis edilməsi bir neçə addımdan ibarətdir:

1.1. Tələblərin təyini: Əvvəlcə, aerport məlumat bazası üçün lazım olan məlumatların nələr olduğu və hansi xüsusiyyətlərinə ehtiyac olduğu belə təyin edilməlidir. Bu, istifadəçilərin hansı məlumatlara çatmaq istəyə biləcəklərini və nələrin avtomatik olaraq toplanması və saxlanılması üçün bir təyinatı müəyyən edəcək.

1.2. Məlumatların cəmlənməsi: Aerport məlumat bazası üçün lazım olan məlumatlar hansı mənbələrdən gəlir və bu mənbələrdən necə cəmləndiriləcəkləri təyin edilməlidir. Bu, məlumatların əldə edilməsi və ardından işlənməsi üçün lazım olan hansı proseslərin nəzərdə tutulmasını daxil edir

1.3. Məlumatların yaradılması: Məlumatlar bir neçə fərqli mənbədən əldə ediləcəyinə görə, onların bir bütün kimi bir yerə saxlanılması lazımdır. Bu, məlumatların bir cədvəldə və ya birdən çox cədvəldə saxlanılması, bütün məlumatların eyni formatda olması və məlumatların həmişə yenilənərək yenilənməsinin təmin edilməsini daxil edir.

1.4. İstifadəçi interfeysinin yaradılması: Aerport məlumat bazası istifadəçilər üçün yaradılmışdır, buna görə də, məlumatların səmərəli istifadəsi və təhlil edilməsi üçün bir istifadəçi interfeysi təmin edilməlidir. Bu, istifadəçilərin məlumatlara əsaslanaraq araşdırmalarını həyata keçirmələrinə imkan verər.

1.5. Təhlil və hesablanmaların təmin edilməsi: Aerport məlumat bazası ilə birgə gələn bir digər addım, məlumatların təhlil edilməsidir. Bu, məlumatların üzərində axtarış edilməsi, filtrlənməsi, müqayisələri və hesablamaları aparılması ilə əlaqəlidir.

1.6. Məlumat bazasının təkmilləşdirilməsi: Məlumat bazası zamanla dəyişə bilər və yenilənə bilər. Buna görə də məlumat bazasının funksiyalarının və təlimatlarının yenilənməsi və təkmilləşdirilməsi vacibdir.

1.7. Məlumat bazasının tərtibi: Tələblər müəyyənləşdikdən və məlumatlar təsis edildikdən sonra, məlumat bazası tərtib edilir. Bu, hər məlumatın nə və ya kim tərəfindən saxlanılacağını və hansı məlumatların hansı cədvəldə saxlanılacağını müəyyənləşdirmək deməkdir. Aeroport məlumat bazasında məlumatların tərtibi, məlumatların effektiv idarə edilməsi və axtarışının sürətli olması üçün əhəmiyyətli bir addımdır. Bu məqsədlə, məlumatlar fərdi cədvəllərdə (tables) saxlanılmalıdır. Məlumat bazası tərtibində, mümkün olan hər bir məlumat tipləri üçün ayrı table yaradılmalıdır. Məsələn, sərnişinlər haqqında məlumatlar və uçuş məlumatları fərqli cədvəllərə sahib olmalıdır. Məlumat bazası tərtibində digər bir

əhəmiyyətli addım isə cədvəllər arasındakı əlaqələrin təyini və təyin edilən əlaqələrin effektiv idarə edilməsidir. Əlaqələr əsasən "əlaqələndirici sütun" (foreign key) vasitəsilə təyin olunur. Bu sütunlar, iki cədvəl arasında əlaqə yaratmaq üçün istifadə olunur. Məsələn, uçuşlar cədvəli və sərnişinlər cədvəli arasında əlaqə yaratmaq üçün, uçuşlar cədvəlində sərnişinlər üçün müəyyən edilən unikal kodu özündə saxlayan əlaqələndirici bir sütun yaradılmalıdır.

1.8. Məlumatların bazaya daxil edilməsi: Məlumatların bazaya daxil edilməsi prosesi təklif edilən tərzə əsasən fərqli ola bilər. Bu proses, proqram təminatları və ya düzəliş edilmiş elektron quraşdırmalar ilə həyata keçirilə bilər. Aerport məlumat bazasında məlumatların bazaya daxil edilməsi, SQL əmrlərinin istifadəsi ilə həyata keçirilir. Daxil ediləcək məlumatlar əsasən formalar üzərindən doldurulur və ya texniki vasitələr vasitəsilə import edilir.

Daxil ediləcək məlumatın hansı cədvələ daxil ediləcəyi təyin edildikdən sonra, INSERT INTO SQL əmrindən istifadə edilərək məlumatlar bazaya daxil edilir. Məsələn:

```
INSERT INTO passengers (first_name, last_name, age, gender, flight_number)
VALUES ('John', 'Doe', 25, 'Male', 'KL123');
```

Bu əmr `passengers` cədvəlində `first_name`, `last_name`, `age`, `gender`, və `flight_number` sütunlarına dəyərlər daxil edir. Əgər cədvəldə mövcud olan bütün sütunlar daxil ediləcəksə, sütunların adları göstərilməyə bilər:

```
INSERT INTO passengers
VALUES ('John', 'Doe', 25, 'Male', 'KL123');
```

Bu halda, SQL məlumat bazası daxilindəki cədvələ daxil ediləcək məlumatların tipi və səhv yoxlanması ilə bağlı qaydalara diqqət etmək vacibdir. Əks halda, bazada səhvli məlumatlar daxil olacaq və bu, məlumatların doğruluğunu və inteqrallığını təhlükə altına salacaqdır.

9. Məlumatların yoxlanılması və təhlili: Məlumatların bazaya düzgün daxil edilməsi əmin olmaq üçün yoxlamaq vacibdir. Məlumatların təhlili və düzəlişi üçün məlumat bazasının istifadəçilər üçün nəzərdə tutulmuş olan qurğuları, nümunələri və prototipləri mövcuddur. Aerport məlumat bazasında məlumatların yoxlanılması və təhlili, məlumatların doğruluğunu və müvafiqliyini yoxlamaq üçün əlavə prosesləri əhatə edir. Məlumatların yoxlanılması prosesi, məlumatların səhvlərini və ya yanlışlıqlarını aşkar etmək üçün istifadə olunan bir seriya proseslərdir. Bu proseslər daxilində, məlumatlar fərdi bir sənəd və ya proseslər daxilindəki digər məlumatlarla müqayisə edilərək təsdiqlənməsi və ya redaktə edilməsi ehtimalı yüksəkdir. Bu proseslər ayrıca, məlumatlar bazasındakı bütün məlumatlarla müqayisə edilərək, tamamilə doğruluq və müvafiqilik təmin edilməlidir. Məlumatların təhlili prosesi, məlumatları daha aşkar və müvafiq bir şəkildə nəzərdən keçirməyi və məlumatları bazada saxlamaq üçün daha uyğun bir formata gətirməyi daxil edir. Bu proses zamanı, məlumatlar üçün səhifələr, siniflər və ya digər məlumatların qruplandırılması istifadə edilir. Bu proses ayrıca, məlumatları sıralamaq, filtrləmək və digər məlumatların həcmi və keyfiyyəti haqqında informasiya əldə etmək üçün əlavə tədbirlər də tələb edə bilər. Aerport məlumat bazasında, məlumatların yoxlanılması və təhlili, məlumatların doğruluğunu və müvafiqliyini təmin etmək üçün çox önəmlidir. Bu proseslər, məlumatların daha dəqiq, tam və ətraflı olaraq sıxlışdırılmasına və məlumat bazasının daha səmərəli və işləyən olmasına kömək edir.

10. Tətbiqetmənin yoxlanılması: Aerport məlumat bazasında tətbiqetmənin yoxlanılması, məlumatların doğruluğunu və düzgünlüyünü yoxlamaq, səhv və problemləri tapmaq və aradan qaldırmaq üçün aparılan prosesdir. Bu proses bir neçə addımdan ibarət ola bilər. Əvvəlcə, məlumatların daxil edilməsi zamanı müvafiq təhlükəsizlik və doğruluq yoxlamaları edilməlidir. Sonra, bazada yadda saxlanan məlumatların düzgünlüyü və doğruluğu müvafiq SQL sorğuları ilə yoxlanılmalıdır. Bu addımda mümkünsə, bazadakı məlumatlarla bərabər olan müqayisə məlumatları da istifadə edilə bilər. Yoxlama zamanı tapılan problemlər aradan qaldırılmalıdır. Bu problemlər məlumatların səhv və ya yanlış daxil edilməsi, tərəfdaşlar və ya müştərilərdən gələn səhvlər və ya bazada olan problemlər ola bilər. Məlumat bazasının düzgün işləməsini təmin etmək üçün tətbiq edilən avtomatik yoxlama və monitorinq sistemləri də mövcuddur. Bu sistemlər əlavə bir təhlükəsizlik tədbiri olaraq fəaliyyət göstərərək bazada ola biləcək problemlərin təxmin edilməsinə və aradan qaldırılmasına kömək edir.

Aerport məlumat bazasının dəyəri.

Aerport məlumat bazasının dəyəri, onun təmin etdiyi faydalı məlumatların miqdarı və keyfiyyətinə və onların istifadə olunması ilə bağlı potensial məhsuldarlığa əsaslanır. Bu məlumatlar hava limanının müxtəlif hissələri arasındakı əlaqələri, müştərilərin və digər tərəfdaşların məlumat tələblərini ödəyir, operasiya proseslərinin təkmilləşdirilməsinə kömək edir və layihələrin və proqramların hazırlanması üçün əsas mənbədir.

Ayrıca, Aerport məlumat bazası, hava limanının ətraf mühitindəki dəyişikliklərə və müştərilərin tələblərinə uyğun olaraq yenilənə bilər və bu da onun dəyərini artırar. Hava limanı, məlumat bazasından istifadə edərək iş proseslərini təkmilləşdirərək, zaman və resursları qənaətli istifadə edərək əməliyyatları daha sürətli və effektiv hala gətirə bilər. Bununla birlikdə, Aerport məlumat bazasının

dəyəri potensial risklərə də əsaslanır. Məlumatların qorunması və gizliliyinin təmin edilməsi üçün uyğun tədbirlərin alınması vacibdir. Məlumatların pozulması, silinməsi, yanlış işlənməsi və ya əlildən düşməsi halında, məlumat bazasının dəyəri azalacaq və hətta potensial risklər təşkil edə bilər.

Biz məlumat verdiyimiz hava limanı məlumat bazasını Python proqramlaşdırma dili vasitəsi ilə yaradacayıq.

Əvvəlcə Python proqramlaşdırma dili haqqında qısa məlumat verək:

Python, Guido van Rossum tərəfindən 1989-cu ildə yaradılmış bir proqramlaşdırma dilidir. Adı, komediyaçı Monty Python-dan gəlir. Python düzgün, oxunaqlı və sadə sintaksisa malikdir. Bu, başlanğıc proqramlaşdırma təhsili üçün ideal olan bir dildir. Həmçinin, güclü bir dil olaraq təsvir edilə bilər, çünki bir çox fərqli sahədə, daxil olmaqla, məlumat elmi, maşın öyrənməsi, web proqramlaşdırması, ağ proqramlaşdırması kimi sahələrdə istifadə olunur. Python, birçox modulla birlikdə gəlir və böyük bir istifadəçi cəmiyyəti vardır. Bu dil, açıq mənbə kodlu və platforma bağlıdır, buna görə də bir çox fərqli platformada işləyə bilər.

Python, geniş fəaliyyət sahələrinə uyğun proqramlaşdırma dili olaraq tanınan yüksək səviyyəli bir proqramlaşdırma dilidir. Əsasən, Python, sadə, açıq və oxunaqlı sintaksisa malikdir. Python dilində yazılmış olan proqramlar, çox asan bir şəkildə oxunur və başa düşülür. Python, çox böyük bir standart kitabxanası ilə birlikdə gəlir və bir çox fəaliyyət sahələrində istifadə olunur - məsələn, məlumat elmi, verilənlər bazası idarəetməsi, məlumatları təhlil etmə, veb proqramlaşdırma və kimi. Python ilə həmçinin, rəqabətli işlər əldə edə bilərsiniz, məsələn, yapay intellekt və maşın öyrənməsinə aid proqramlar, verilənlər bazası tətbiqləri, və web proqramlaşdırma tətbiqləri kimi. Python, interaktiv interpreteri

ilə birlikdə gəlir və bu, dilin öyrənilməsinə və izlənməsinə kömək edir. Bu özəllik, kodunuzun səhv tapılması zamanı sizi mövqeyinə düzgün həll tapmağa imkan verir.

Python ilə, interaktiv interpreter vasitəsilə məlumatları sınamaq və dəqiqliklə test etmək üçün asan bir şəkildə kod yazmaq və çalışdırmaq mümkündür. Həmçinin, Python dilində, veb və proqram təminatları ilə işləmək üçün də bir çox fərdi və ya hazırda mövcud olan modullar var. Python, öyrənməsi üçün asan bir proqramlaşdırma dilidir və bu, təcrübəsi az olan proqramçılar üçün çox uyğundur. Məsələn, sintaksis qaydalarının sadəliyi və dəqiqliyi, öyrənməsi asan və bir çox səviyyələrdə bir çox kitab və tutorial mövcuddur.

Biz bu, hava limanı məlumat bazsını yaratmaq üçün Python proqramlaşdırma dilinin kitabxanalarından istifadə edəcəyik. Python kitabxanaları, hazırda yazılmış funksiyalar, modullar və obyektlər daxil edən, kod yazmağı sürətləndirən və təkrarlanmayan kodu yenidən istifadə edərkən vaxt və ehtiyacı azaltan proqramlar üçün istifadə olunur. Kitabxanalar proqramçıların lazımi olan funksiyaları və ya həlləri yenidən icad etməmələrini təmin edir.

2.1. Python kitabxanaları müxtəlif məqsədlər üçün mövcuddur, məsələn:

- Məlumatlar bazası əməliyyatları (SQLAlchemy, Psycopg2, pyodbc)

- Rəqəmsal hesablama və statistik analiz üçün (NumPy, Pandas, SciPy, Statsmodels)

- Təsvir və görsəlləşdirilmə üçün (Matplotlib, Seaborn, Plotly)

- Maşın öyrənməsi və incəsənətli analiz üçün (TensorFlow, Keras, PyTorch, Scikit-learn)

- Şəkil işləməsi və yaradıcılıq üçün (Pillow, Pygame, Turtle)

- və daha bir çox məqsədlər üçün lazım olan kitabxanalar mövcuddur.

Python proqramlaşdırma dili, çeşitli SQL məlumat bazası sistemləri ilə birləşdirilə bilən bir çox kitabxana və modullar təklif edir. Bu kitabxanalar arasında aşağıdakılar var:

1. SQLite3: Python proqramlaşdırma dilinin ən populyar və asanlıqla istifadə edilən SQL məlumat bazası kitabxanasıdır. Bir neçə sətirdən ibarət sadə bir kodla bu kitabxanadan istifadə edərək SQLite3 məlumat bazasında əməliyyatlar həyata keçirə bilərsiniz.

2. SQLAlchemy: Bu kitabxana, fərqli SQL məlumat bazası sistemləri ilə işləmək üçün yüksək səviyyəli bir arayüz təklif edir. Bu, fərqli məlumat bazası sistemlərində əməliyyatlar yerinə yetirmək üçün birbaşa SQL əmrinə ehtiyac duymadan proqramlarınızı yazmağa imkan verir. SQLAlchemy, Python üçün açıq mənbə kodlu bir SQL Toolkit və ORM (Object-Relational Mapping) kitabxanasıdır. Bu kitabxana, Python proqramçılarının fərqli məlumat bazaları ilə işləməsini asanlaşdırır və ORM vasitəsilə məlumat bazası ilə obyekt tərəfli interfeys yaradır. SQLAlchemy, müxtəlif SQL məlumat bazaları (MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite və s.) ilə uyğun gəlir və birbaşa SQL yazmağı və ya ORM vasitəsilə Python obyektlərini məlumat bazası sətirlərinə birləşdirməyi mümkün edir. ORM vasitəsilə, məlumat bazasına yazmaq və oxumaq, məlumatları dəyişdirmək və silmək və məlumatlar arasında əlaqələr qurmaq daha asan və müxtəlif məlumat bazaları ilə uyğun gəlir. SQLAlchemy, kompleks SQL sorğuları və cədvəllər üçün müxtəlif dərəcəli təkmilləşdirilmiş interfeyslər təqdim edir və ayrıca həddindən artıq SQL injection hücumlarına qarşı qoruma təmin edir. Bunların hamısı, SQLAlchemy-nin sərbəst və açıq mənbəli olması və bir çox digər faydalı kitabxanalarla uyğunlaşması ilə əlaqələndirilir.

3. Psycopg2: Bu PostgreSQL məlumat bazası ilə işləmək üçün kitabxanadır. PostgreSQL-da SQL əmrləri yerinə yetirmək üçün ən populyar və təhlükəsiz seçimdir. Psycopg2, PostgreSQL verilənlər bazası ilə Python proqramlaşdırma dili vasitəsi ilə əlaqə qurmaq üçün istifadə edilən bir kitabxanadır. Bu kitabxana, PostgreSQL serveri ilə əlaqə yaratmaq, SQL əmrini yerinə yetirmək və verilənlər bazasından məlumat almaq üçün təchizat təmin edir. Psycopg2 kitabxanası, PostgreSQL verilənlər bazasına sürətli və effektiv əlaqə yaratmaq üçün tənzimlənə bilər və açıq mənbəli olduğu üçün geniş istifadəçi cərgəsinə malikdir. Bu kitabxana, həm də Python dilində yazılmış dairəvi ilə MySQL və SQLite kimi digər verilənlər bazaları ilə də uyğunlaşdırılabilir.

4. PyMySQL: Bu MySQL və MariaDB məlumat bazası sistemləri ilə işləmək üçün kitabxanadır. Bu, MySQL məlumat bazasını istifadə edərək veb tətbiqləri, oyunlar və ya hər hansı bir proqramın hazırlanması üçün istifadə edilə bilər. PyMySQL, Python dilində MySQL serverləri ilə əlaqə qurmaq üçün istifadə edilən bir Python kitabxanasıdır. Bu kitabxana, MySQL serverləri ilə əlaqə qurmanıza və SQL əmrləri yerinə yetirməyə imkan verir. PyMySQL, MySQL serverləri ilə etibarlı və sürətli əlaqələr yaratmaq üçün sadə və asan bir interfeys təqdim edir. Bununla birlikdə, PyMySQL kimi kitabxanaların SQL injection hücumlarına qarşı təhlükəsizlik məsələlərini nəzərə alaraq düzgün istifadəsi çox vacibdir.

5. cx_Oracle: Bu Oracle verilənlər bazası ilə işləmək üçün kitabxanadır. Bu, Oracle verilənlər bazasında SQL əmrlərinin yerinə yetirilməsi, yaratılması və xüsusi əmrlərin icrası üçün istifadə edilir. cx_Oracle, Oracle verilənlər bazası ilə Python arasında bağlantı qurmaq üçün istifadə edilən bir Python kitabxanasıdır. cx_Oracle kitabxanası, Oracle Database istifadə edən hər hansı bir tətbiqetmə proqramı üçün Python dilində ilətişim qurmağı mümkün edir. cx_Oracle

kitabxanası, Oracle verilənlər bazasına SQL ifadələri göndərmək və alınan cavabları işləmək üçün əlverişlidir. cx_Oracle istifadə edərək Python vasitəsilə Oracle verilənlər bazasına daxil olan məlumatları oxuya, yazıya və ya dəyişə bilərsiniz. cx_Oracle, özünü yüksək performans və stabillik ilə fərqləndirir. Oracle verilənlər bazası ilə asan və təhlükəsiz əlaqə qurmaq üçün istifadə edilir.

Bu sadəcə bir neçə Python SQL məlumat bazası kitabxanasıdır. Python-da daha bir çox SQL məlumat bazası kitabxanası mövcuddur. İşiniz üçün ən uyğun olanını seçmək üçün SQL məlumat bazası növünüzdən və tələblərinizdən asılı olaraq bu kitabxanalar arasında bir seçim edə bilərsiniz.

Bu, kitabxanalar arasından bizə lazım olacaq kitabxana məlumatlar bazası əməliyyatlarını aparmaq üçün istifadə olunan **SQLlite** kitabxanasıdır. SQLite, server olmadan çalışan, özündən öyrənilməsi çox asandır və özləri tərəfindən idarə olunan bir SQL məlumat bazası kitabxanasıdır. Bu, hər hansı bir serveri idarə etmək üçün istifadə olunmayan, yerli bir SQL məlumat bazasıdır. Python SQLite, Pythondan SQL məlumat bazalarına məlumat yazmaq və oxumaq üçün istifadə edilən standart bir moduldur. Bu, sadə və istifadəsi çox asan bir moduldur və hər hansı bir proqramda SQLite məlumat bazasına əlavə funksional qoymaq üçün istifadə oluna bilər. Python SQLite modulu ilə əməliyyatlar, SQL məlumat bazasında cədvəllər yaratmaq, cədvəlləri dəyişdirmək, məlumat yazmaq və oxumaq, məlumat bazasından məlumat silmək və s. kimi bir sıra əməliyyatları icra edə bilərsiniz.

Məlumat bazaları zamanla inkişaf edir və bir sıra problemlər yaradır. Bu səbəbdən ötəri, əksər məlumat bazaları sıxlığı azaltmaq üçün şəkillərin saxlanılmasına icazə vermirlər. Bununla birlikdə, SQLite məlumat bazası proqramı bu imkanı təqdim edir. SQLite, yüngül sıxlığı olan bir məlumat bazası

olduğu üçün, əlaqələndirilmiş fayllar və ya bəzi olaraq şəkillərin saxlanılması üçün ideal bir seçimdir. Şəkillər məlumat bazasında base64 şəklində saxlanırlar.

Hava limanı məlumat bazsının Python proqramlaşdırma dilində yazılmış kod hissəsi ilə tanış olaq. Bu kodlar vasitəsi ilə biz hava limanın da qeydiyyatdan keçən və uçuş edən hər bir kəsin məlumatlarını bir bazaya yığıb saxlaya bilərik. Bu məlumatları yadda saxlamaq üçün bizə sərnişinlər qeydiyyatdan keçərkən təqdim etdikləri məlumatlar lazımdır. Həmin məlumatlardan istifadə edərək biz məlumatları avtomatlaşdırılmış şəkildə və ya əl ilə hava limanı məlumat bazasına daxil edə bilərik. Bu məlumatların bazada saxlanılması çox böyük önəm daşıyır.

Dəyərli məlumatların məlumat bazasında saxlanılması həqiqətən çox böyük önəm daşıyır. Hava limanı məlumat bazasında saxlanılan məlumatlar sərnişinlər, uçuşlar, bilet məlumatları, yük məlumatları, texniki məlumatlar və daha çox informasiya kimi çeşidli dəyərli məlumatları əhatə edə bilir. Bu məlumatlar hava limanının günlük fəaliyyətinin yekunları, strateji qərarların verilməsi, təhlil və hesablama işləri, idarəetmə prosesləri, inteqrasiya ilə digər təşkilatlar arasında məlumat mübadiləsi və daha çox üçün istifadə oluna bilər.

Bu məlumatların böyük bir dəyəri olduğundan, məlumat bazasının təhlükəsizliyi, gizliliyi və mövcudluğu kritikdir. Təhlükəsizlik tədbirləri, şifrələmə, istifadəçi yetkiləndirməsi, günlük jurnalizasiya və yedəkləmə kimi texniki tədbirlər vasitəsilə məlumatların qorunması təmin edilməlidir. Məlumat bazasının mövcudluğu üçün yedəkləmə prosedurları, növbəti yedəkləmə sistemləri və məlumat itirməyə qarşı planlar qurulmalıdır.

Hava limanı məlumat bazasındakı dəyərli məlumatların doğru və düzgün şəkildə saxlanılması, məlumat bazasının effektiv işləməsi, dəyişikliklərin izlənilməsi və inteqrasiya mövqeyinin qorunması üçün də düzgün tənzimlənməsi və təhlil

edilməsi vacibdir. Bu, məlumatların səhifələnməsi, indekslənməsi, optimallaşdırılması və performansın yaxşılaşdırılması ilə əlaqəlidir.

Əvvəlcədən planlama, uyğun təhlükəsizlik tədbirləri və effektiv idarəetmə, hava limanı məlumat bazasında saxlanılan dəyərli məlumatların sürətli, təhlükəsiz və effektiv şəkildə işləməsinin təminatını verir.

Məlumat bazasının effektiv işləməsi, məlumatların sürətli və effektiv şəkildə yığılması, saxlanılması, axtarılması və təhlil edilməsi deməkdir. Effektiv işləmək, performansın yaxşılaşdırılması və məlumat bazasının istifadəçilərə sürətli cavab verə biləcək şəkildə fəaliyyət göstərməsi anlamına gəlir.

3. Aşağıdakı məlumatlar, məlumat bazasının effektiv işləməsini təmin etmək üçün əhəmiyyətli olan bəzi məsələləri təsvir edir:

3.1. İndeksləmə: İndekslər, məlumat bazasının tərkibindəki məlumatların sürətli axtarışını təmin etmək üçün istifadə olunur. İndeksləmə, məlumatların sıralanması və axtarışa hazırlanması deməkdir. Məlumat bazasında uyğun indekslər yaradılaraq axtarış əməliyyatları sürətləndirilir.

3.2. Optimallaşdırma: Hava limanı məlumat bazasında optimallaşdırma, performansı və effektivliyi artırmaq üçün məlumat bazasının strukturu, indekslər, sorğular və digər elementlərin təkmilləşdirilməsi prosesidir. Optimallaşdırma, məlumat bazasının performansını artıraraq daha sürətli və effektiv işləyən bir sistem yaratmağa kömək edir.

Aşağıdakı sahələrdə optimallaşdırma işləri hava limanı məlumat bazasında tətbiq edilə bilər:

1. Optimallaşdırılmış sorğular: Sorğuların effektivliyini artırmaq üçün uyğun indekslər və effektiv sorğu strukturunu istifadə etmək lazımdır. Sorğuların yavaş olmasına səbəb olan məsələlər identifikasiya edilməli və təkmilləşdirilməli.

3. Tənzimləmə və təmir: Məlumat bazasında düzgün tənzimləmələr və təmir işləri həyata keçirilməlidir. Tənzimləmə işləri zamanı məlumat bazasının performansını artırmaq üçün müxtəlif təkmilləşdirmələr edilir.

6. Server konfiqurasiyası: Server konfiqurasiyası, bir serverin çalışma parametrlərini və ayarlarını tənzimləmə mənbəyi ilə bağlıdır. Bu ayarlar, serverin performansını, təhlükəsizliyini, istifadəçi qəbulunu və digər funksiyalarını tənzimləməyə kömək edir. Hava limanı məlumat bazasının effektiv işləməsi üçün aşağıdakı server konfiqurasiyası aspektlərində diqqət yetirilməlidir:

1. Yaddaş ayarları: Məlumat bazası üçün ayrılan yaddaş miqdarı optimal olmalıdır. İyirmi konfigurasiya parametri kimi bilinən nöqtədən etibarən, məlumat bazasının yüklənmə və sorğu icrası üçün lazım olan minimal və maksimal yaddaş həcmi təyin edilməlidir.

2. İş parçacığı və hərəkət ayarları: Hərəkət növləri və iş parçacığı limitləri ilə əlaqəli ayarlar, məlumat bazasının yüklənmə və sorğu icrası səhifələri üçün lazım olan optimal performansı təmin etməyə kömək edir.

3. İndeks və statistiklər: İndeks və statistiklər ayarları, sorğuların daha sürətli icrasını və məlumat bazasının təhlükəsizliyini təmin etməyə kömək edir. İndeks parametrləri tənzimlənməlidir və statistiklər yenilənməlidir.

4. Protokollar və təhlükəsizlik: Serverin istifadə etdiyi protokollar və təhlükəsizlik ayarları məlumat bazasının təhlükəsizliyini təmin etmək üçün əhəmiyyətli olduğu kimi, performansı da təsir edir. Protokolların və təhlükəsizlik

ayarlarının optimal şəkildə tənzimlənməsi məlumat bazasının effektiv işləməsinə kömək edir.

5. İştirakçı və bələdçi konfiqurasiyası: İştirakçı və bələdçi serverlərin uyğun konfiqurasiya edilməsi, məlumat bazasının performansını artırmağa və yüklənmələri məhdudlaşdırmağa kömək edir. Bu, serverin yüklənməni bölüşmək və yedəkləməni tənzimləmək üçün əhəmiyyətli olan ayarlarını daxil edir.

Bu yalnız bir neçə örnək olaraq verilmiş server konfiqurasiyası aspektləridir. Hava limanı məlumat bazasının optimal performansı təmin etmək üçün, hər bir konkret hallara uyğun olaraq server konfiqurasiyasının tənzimlənməsi və təkmilləşdirilməsi əhəmiyyətli rol oynayır.

3. Səhifələmə: Hava limanı məlumat bazasında səhifələmə, həcmi böyük olan məlumatların təqdimatını və idarəetməsini daha effektiv və effektiv bir şəkildə həyata keçirmək üçün istifadə olunur. Bu, məlumat bazasında sayca çox məlumatların böyük hissələrə bölünüb səhifələrə bölməsi və bu səhifələrin tək-tək yüklənməsi deməkdir

Səhifələmə prosesi ümumiyyətlə limit və ofset dəyərləri istifadə edərək həyata keçirilir. Limit dəyəri, bir səhifədə yüklənəcək maksimum məlumat sayını təyin edir, məsələn, 20 məlumat. Ofset dəyəri isə hansı məlumatdan başlanacağını göstərir, məsələn, ilk səhifədən başlanılacaqsa ofset dəyəri 0 olur. İstifadəçi səhifələmə funksiyasını istifadə edərək hər bir səhifədəki məlumatları ala bilər. Səhifələmə prosesi, istifadəçiyə yalnız lazım olan məlumatları təqdim edir və təqdim olunan məlumatların həcmindən asılı olaraq məlumat bazasının performansını artırır. Səhifələmənin əsas faydaları məlumat bazasının yüklənmə vaxtını azaldır, server tərəfdən təşkil olunan təpki vaxtını artırır və mövcud server resurslarını effektiv şəkildə istifadə edir.

4. Cədvəl dizaynı: Cədvəl dizaynı, bir məlumat bazasında cədvəllərin və onların atributlarının optimal və səmərəli şəkildə tənzimlənməsini və təmsil edilməsini ifadə edir. Məlumat bazasında doğru cədvəl dizaynı, məlumatların effektiv saxlanılması, işlənməsi və məlumat bazası təhlili üçün əhəmiyyətlidir.

Cədvəl dizaynı zamanı aşağıdakı məsələlərə diqqət yetirilməlidir:

4. İlişkiləndirmə (Normalization): İlişkiləndirmə (Normalization), məlumat bazasında cədvəllərin optimal şəkildə tənzimlənməsi prosesidir. Bu prosesdə məlumatlar düzgün şəkildə cədvəllərə bölmək və cədvəllər arasında münasibətlər yaratmaq hədəfə çatır. İlişkiləndirmənin əsas məqsədi, məlumatların təkrarlanmasını azaldaraq inkonsistentlikləri aradan qaldırmaqdır. Bu, məlumat bazasının effektiv idarə olunmasını, məlumatların effektiv saxlanılmasını və təhlil edilməsini təmin edir.

İlişkiləndirmə prosesində aşağıdakı normalizasiya səviyyələri istifadə olunur:

4.1. Birinci normalizasiya (1NF): Birinci normalizasiya (1NF), məlumat bazasında inkonsistentlikləri və təkrarlanan məlumatları aradan qaldırmaq üçün tətbiq edilən bir ilişkiləndirmə prinsipidir. Birinci normalizasiya, cədvəllərin atomar dəyərlərlə bölünməsini və hər bir sətirdə yalnız bir dəyər qrupunun olmasını tələb edir.

1NF tələbləri aşağıdakı prinsiplərə əsaslanır:

1. Cədvəllər atomar dəyərlərlə bölünməlidir: Bir cədvəlin hər bir sütunu, məlumatları atomar dəyərlərə bölüşdürməlidir. Bu deməkdir ki, bir sütunda birdən çox dəyər olmamalıdır.

2. Hər sütunda tək tip məlumatlar olmalıdır: Hər bir sütun yalnız bir məlumat tipinə aid olmalıdır. Bu, cədvəllərdə eyni tipdə məlumatların qruplaşdırıldığı anlamına gəlir.

3. Hər sətr unikal bir identifikatora malik olmalıdır: Hər bir sətr, cədvəlin identifikator sütunu (primar açar) vasitəsilə fərdi bir şəkildə müəyyənləşdirilməlidir. Bu, hər bir sətrin tək və unikal olmasını təmin edir.

Birinci normalizasiya, məlumat bazasında təkrarlanan məlumatların azaldılmasına və məlumatların daha düzgün, təhlükəsiz və effektiv şəkildə saxlanılmasına imkan verir. Bu prinsiplər, məlumat bazasının uyğun dizayn edilməsi, verilənlərin tutarlılığının və məlumatların doğruluğunun təmin edilməsi üçün əhəmiyyətlidir.

4.2. İkinci normalizasiya (2NF): İkinci normalizasiya (2NF), məlumat bazasında təkrarlanan məlumatları aradan qaldırmaq və cədvəlləri daha da təkmilləşdirmək üçün tətbiq edilən bir ilişkiləndirmə prinsipidir. 2NF, birinci normalizasiyadan istifadə edir və 1NF-nin tələblərinə əlavə olaraq birbaşa olmayan funksional dənətləmələri aradan qaldırmağı hədəfləyir.

2NF-nin əsas tələbləri aşağıdakılardır:

1. İdentifikatora tam asılı olmalıdır: Hər bir cədvəl içində, cədvəlin bütün sütunları identifikatora tam asılı olmalıdır. Bu deməkdir ki, bir sətrdəki hər bir sütun, identifikatorun tamamilə təyin edicisi olmalıdır və digər sütunların dəyərlərini müəyyən etməlidir.

2. İndependent altcədvəllərə bölünməlidir: Cədvəl, hər hansı bir sütunda yalnız identifikatorla əlaqəli olan məlumatları içərməlidir. Ehtimal edilən altcədvəllər yaradılmalı və bu altcədvəllər özünəməxsus identifikatorlara malik olmalıdır.

İkinci normalizasiya, cədvəllərin daha da təkmilləşdirilməsini və məlumatlar arasında funksional əlaqələrin düzgün şəkildə təyin edilməsini təmin edir. Bu, məlumatlar arasında təkrarlanan dəyərlərin azalmasını və məlumat bazasının strukturu və performansı üçün daha optimal bir düzəlişin əldə edilməsini təmin edir.

4.3. Üçüncü normalizasiya (3NF): Üçüncü normalizasiya (3NF), məlumat bazasında təkrarlanan məlumatları aradan qaldırmaq, cədvəlləri daha da təkmilləşdirmək və funksional dənətləmə problemlərini həll etmək üçün tətbiq edilən bir ilişkiləndirmə prinsipidir.

3NF-nin əsas tələbləri aşağıdakılardır:

1. Birinci və ikinci normalizasiya qaydalarına əməl edilməlidir: Cədvəl, 1NF və 2NF qaydalarını ödəməlidir. Bu deməkdir ki, hər bir sütun yeganə dəyər saxlamaq üçün yaradılmış olmalıdır və altcədvəllər identifikatorla əlaqəlidir.

2. İndependent, transitive funksional dənətləmələrdən azad olmalıdır: Cədvəl, funksional dənətləmələrdən azad olmalıdır. Bu deməkdir ki, identifikatora qoşulan hər hansı bir sütun, yalnız identifikatorun dəyərini təyin etməlidir və digər sütunlarla funksional əlaqələr qurmaq üçün bu sütunlardan asılı olmamalıdır.

Üçüncü normalizasiya, məlumat bazasının daha da təkmilləşdirilməsini və məlumatlar arasında funksional əlaqələrin daha düzgün şəkildə qurulmasını təmin edir. Bu, məlumat bazasının əsaslı və optimallaşmış bir struktura sahib olmasını və performans problemlərinin azalmasını təmin edir. Ayrıca, verilənlərə əsaslanan anomalilərin, təkrarlanan məlumatların və informasiya itki ilə bağlı problem əlamətlərinin azalmasına kömək edir. Bu normalizasiya səviyyələri, məlumat bazasının təhlili və təyini üçün əhəmiyyətlidir. İlişkiləndirmə

prinsiplərinə əsaslanan cədvəl dizaynı, məlumatların səmərəli şəkildə saxlanılmasını, dəyişdirilməsini və axtarışını təmin edir. Həmçinin, məlumat bazasının təhlil edilməsi və digər proseslər üçün optimal mühit yaradır.

2. Attributlar və məlumat tipləri: Attributlar məlumat bazasında saxlanılan məlumatların təmsil etdiyi xüsusiyyətlərdi. Bir cədvəldə hər bir sütun bir atributu təmsil edir. Veri tipləri isə bu atributların hansı növ veriləri saxladığını göstərir.

Bəzi ən çox istifadə olunan veri tipləri aşağıdakılardır:

1. Integer (tam ədəd): Integer, tam ədədləri ifadə etmək üçün istifadə olunan bir veri tipidir. Python-da integer dəyərləri düz tam ədəd dəyərləri olaraq ifadə olunur. Tam ədədlər müsbət, mənfi və ya sıfır dəyər ola bilər. Python tam ədədləri ədədə uyğun olaraq təyin edir və riyazi əməliyyatları üçün istifadə olunur.

Python-da tam ədəd tipində dəyişənlər ədəd təyin edə bilir və ya ədəd əməliyyatları aparmaq üçün istifadə oluna bilər:

```
x = 5
y = -10
z = 0

# Riyazi əməliyyatlar
a = x + y
b = x * z
c = y / x

print(a)  # Çıxış: -5
print(b)  # Çıxış: 0
print(c)  # Çıxış: -2.0
```

Bu nümunədə, əvvəlcə tam ədəd dəyişənlərinə dəyərlər təyin edilir. Sonra riyazi əməliyyatlar ilə dəyişənlər arasında əməliyyatlar aparılır və nəticələr çıxarılır. Riyazi əməliyyatlarda toplama (+), vurma (*), bölmə (/) və s. əməliyyatlar istifadə olunur. Tam ədədlər riyazi əməliyyatlar, şərt ifadələri, dövri operatorlar və s. kimi çeşitli proqramlaşdırma təlimatlarında istifadə olunur.

2. Float (hərəkətli nöqtəli ədəd): Float, hərəkətli nöqtəli ədədləri ifadə etmək üçün istifadə olunan bir veri tipidir. Python-da float dəyərləri onluq sistemində hərəkətli nöqtəli ədədlər olaraq ifadə olunur. Float dəyərləri, bir tam hissə və bir dəqiqlik hissəsi olan rəqəmlərdən ibarətdir. Əksər hallarda rəqəmlər arasında nokta işarəsi ilə ayrılır. Float dəyərləri ədədi əməliyyatlarda və riyazi funksiyaların tətbiqində istifadə olunur.

Python-da float dəyərləri təyin etmək üçün onluq sistemində hərəkətli nöqtəli rəqəmlər istifadə olunur:

```
x = 3.14
y = -2.5
z = 0.0

# Riyazi əməliyyatlar
a = x + y
b = x * z
c = y / x

print(a)  # Çıxış: 0.64
print(b)  # Çıxış: 0.0
print(c)  # Çıxış: -0.7961783439490446
```

Bu nümunədə, hərəkətli nöqtəli dəyişənlərə dəyərlər təyin edilir və sonra riyazi əməliyyatlar aparılır. Float dəyərləri ilə toplama (+), vurma (*), bölmə (/) və s. riyazi əməliyyatlar aparmaq mümkündür. Float dəyərləri dəyişənlərin yanı sıra

funksiyonların qaytaracağı nəticələrdə, çeşitli bilimsel və istatistik məlumatlarda, grafik tərtibində və s. kimi çeşitli yerlərdə istifadə olunur.

3. String (simvol cədvəli): String, yəni mətn veri tipi, simvolların və mətnlərin saxlanılması üçün istifadə olunan veri tipidir. Python və digər proqramlaşdırma dillərində bu veri tipi str adı ilə ifadə olunur. Stringlər, ardıcıllıq (sequence) kimi işləyir, yəni bir-birinə ardıcıl olaraq düzəldilmiş simvollar topluluğunu ifadə edir. Hər bir simvol bir indeksə malikdir və indekslər 0-dan başlayır. Mətnlər tək və ya çift qoşa işarələrlə (', ") təyin edilərək ifadə olunur.

Python-da stringlər dəyişdirilə bilməz (immutable) olduğundan, bir dəyişənə təyin edildikdə dəyişdirilməz. Lakin, stringlər üzərində bir çox əməliyyatlar aparılabilər, məsələn, birləşdirilmə, kəsmə, bölmə, çevirmə, indeksləmə və s. Bu əməliyyatlar stringlərin manipulyasiyasını və dəyişənlərdə saxlanılmasını təmin edir.

Nümunə olaraq, aşağıdakı kod hissəsi bir stringin dəyərini təyin edir və stringin uzunluğunu çap edir:

```
mehsul_adı = "Telefon"
uzunluq = len(mehsul_adı)
print(uzunluq)  # Çıxış: 7
```

Bu kod hissəsində "Telefon" adlı bir string dəyişəni təyin edilir və len() funksiyası ilə bu stringin uzunluğu hesablanır. Əldə olunan nəticə 7 çıxış olaraq göstərilir, çünki "Telefon" ifadəsində 7 simvol var.4. Date (tarix): Tarixləri təmsil edir. İstifadəçilərə tarixi (il, ay, gün) saxlamaq imkanı verir. Məsələn, 2021-09-30 kimi.

5. Time (vaxt): Vaxtları təmsil edir. İstifadəçilərə saat, deqiqə, saniyə kimi vaxt dəyərlərini saxlamaq imkanı verir. Məsələn, 12:30:45 kimi.

6. Boolean (Boolean): Boolean, iki dəyərdən ibarət olan bir veri tipidir: doğru və yanlış. Python-da bu dəyərlər True (doğru) və False (yanlış) olaraq ifadə olunurlar.Boolean dəyərləri çoğu zaman şərt ifadələrində, loqika əməliyyatlarında və yoxlamalarda istifadə olunur. Şərt ifadələri, proqramın müxtəlif şərtlərə görə fərqli davranışlar sergileməsinə imkan verir. Loqika əməliyyatları isə boolean dəyərlər arasında əməliyyat aparmağa imkan verir.

Python-da boolean dəyərlər hesablanmış şərt ifadələrindən və loqika əməliyyatlardan alınır. İstifadəsi çox sadədir:

```
x = 5
y = 10
# Şərt ifadəsi
if x < y:
    print("x, y-dan kiçikdir.")  # Çıxış: x, y-dan kiçikdir.

# Loqika əməliyyatı
z = x < y and x > 0
print(z)  # Çıxış: True
```

Bu nümunədə, əvvəlcə x və y dəyişənlərinə dəyərlər təyin edilir. Ardından şərt ifadəsi ilə x-dəki dəyərin y-dəkindən kiçik olub olmadığı yoxlanılır və uyğun olarsa şərt ifadəsinə daxil olan blok icra olunur. Loqika əməliyyatında isə x-dəki dəyərin y-dən kiçik olma və x-dəki dəyərin 0-dan böyük olma şərti təqdim olunur və nəticə olaraq z dəyişəninə True dəyəri təyin edilir.

Bu yalnız bir neçə veri tipi nümunəsidir. Python və digər proqramlaşdırma dillərində daha çox veri tipi mövcuddur və hər biri fərqli məlumatları saxlamağa və əməliyyatlarını aparğa imkan verir. Hər bir məlumat bazası idarəçisi də bu veri tiplərini dəstəkləyir və uyğun veri tiplərini seçməklə məlumatların doğru şəkildə saxlanılmasını təmin edir.

3. Primar və xüsusiyyət açarları: Primar və xüsusiyyət açarları, verilənlər bazasında məlumatların sıralanması və bir-birilərinə bağlanması üçün istifadə olunan konseptlərdən biridir.

- Primar açar (Primary Key): Primar açar, hər bir məlumat sətirinin unikal və təyin edici bir kimliyə malik olan sütunudur. Məlumat bazasında primar açar vasitəsilə hər bir sətirə unikal bir identifikator (ID) təyin edilir. Primar açar, hər bir sətiri fərqli və təyin edici edir və məlumatların hədəfli və sürətli axtarışında əhəmiyyətli rol oynayır. Ən çox istifadə olunan primar açar veri tipi integer və ya stringdir.

- Xüsusiyyət açarları (Attribute Keys): Xüsusiyyət açarları, verilənlər bazasındakı sətirləri digər sütunlar vasitəsilə bir-biri ilə əlaqələndirən sütunlardır. Bu açarlar vasitəsilə hər bir sətirin başqa bir sətir ilə əlaqəsi təmsil edilir. Xüsusiyyət açarları, sətirləri bir-biri ilə əlaqələndirmək və məlumatların birləşdirilməsini təmin etmək üçün istifadə olunur. İstifadə olunan xüsusiyyət açarları təyin edilmiş əlaqələrə görə dəyişə bilər. Primar və xüsusiyyət açarları verilənlər bazasında məlumatların düzgün şəkildə təyin edilməsini və əlaqələndirilməsini təmin edir. Bu açarlar vasitəsilə verilənlər bazasındakı məlumatlar effektiv şəkildə axtarışa və birləşdirməyə imkan verir.

4. Əlaqələr və referensial əlaqələr: Əlaqələr və referensial əlaqələr, verilənlər bazasında fərqli cədvəllər arasında münasibətlərin təyin edilməsi və təmin edilməsi üçün istifadə olunan mekanizmalardır. Əlaqələr, fərqli cədvəllər arasında bağlantı yaratmağa imkan verir və məlumatların daha effektiv və düzgün şəkildə saxlanılmasını təmin edir. Bu sayədə məlumatlar bir-biri ilə əlaqələndirilərək məlumatlar bazasının bütünlüyü və konsistentliyi qorunur.

Referensial əlaqələr (Referential Integrity), əlaqəli cədvəllər arasında məlumatın əlaqələndirilməsində müsbət təsir təmin edir. Bu əlaqələr, məlumatların əlaqəli cədvəllərdəki münasibətlərə uyğun olaraq saxlanılmasını və müvafiq əməliyyatların (sil, güncəllə, əlavə etmə, kəs) təyin edilmiş qaydalar üzrə yerinə yetirilməsini təmin edir. Əlaqələr və referensial əlaqələr, verilənlər bazasının bərabərliyini, məlumatların doğruluğunu, bütönlüyünü və konsistentliyini təmin etmək üçün əsas məkanizmalardır. Bu, məlumatların səmərəli idarə edilməsinə və verilənlər bazasının effektiv və məqsədəuyğun işləməsinə imkan verir.

Cədvəl dizaynı məlumat bazasının effektiv işləyərək məqsədlərə cavab verən və performansını artıran əsas hissəsidir. Bu prinsiplərə uyğun şəkildə cədvəllərin təsis edilməsi və tənzimlənməsi, məlumat bazasının effektiv və səmərəli şəkildə istifadə edilməsinə imkan verir.

5. Yedəkləmə və geri qaytarılma: Yedəkləmə və geri qaytarılma (Backup and Recovery), verilənlər bazasında saxlanılan məlumatların qorunması və ehtiyatlılıq məqsədləri ilə istifadə olunan bir prosesdir. Yedəkləmə, verilənlər bazasında olan məlumatların bir neçə nüsxəsinin hazırlanması deməkdir. Bu, əsas verilənlər bazasının fiziki və ya məntiqi şəklində kopyasının yaradılması anlamına gəlir. Yedəkləmə prosesi, verilənlər bazasında oluşan məlumat itkilərinə qarşı təhlükəsizlik təmin edir. Yedəkləmə prosesi, müvafiq plan və strukturla təyin olunmalıdır və düzgün intervalda təkrarlanmalıdır.

Geri qaytarılma (Recovery), yedəklənmə prosesində hazırlanan məlumatların istifadə edilməsi və əsas verilənlər bazasının itkin məlumatların bərpa edilməsi üçün geri yüklənməsini ifadə edir. Bu proses, məlumat itkisinin, xətalı məlumatların və ya verilənlər bazası problemlərinin düzəltilməsi məqsədi ilə həyata keçirilir. Geri qaytarılma prosesi, yedəkləmə nüsxələrinin düzgün şəkildə saxlanıldığından və uyğun protokollara əsaslanaraq icra edildiyindən əmin

olmağı tələb edir. Yedəkləmə və geri qaytarılma prosesləri, verilənlər bazasının məlumatların itkisinə və potensial fəlakətlərə qarşı təhlükəsizliyini təmin edir. Bu proseslər, işlənən sistemə uyğun tənzimlənmiş və test edilmiş yedəkləmə və geri qaytarılma siyasətinə əsaslanmalıdır. Yedəkləmə nüsxələrinin müvafiq saxlanma yerlərində təhlükəsiz və məsuliyyətli şəkildə qorunması və geri qaytarılma proseslərinin düzgün şəkildə idarə edilməsi, verilənlər bazasının mövcudluğunu və davamlılığını təmin edir.

6. Cədvələr arası inteqrasiya: Əgər aerport məlumat bazası digər sistemlərlə, məsələn, hava yolçuluğu idarəetmə sistemləri, bilet satış sistemləri və s. ilə əlaqəli olarsa, bu sistemlərlə inteqrasiya təhlükəsiz və səmərəli bir şəkildə olmalıdır. Bu, məlumatların düzgün şəkildə paylaşılmasını və təhlili üçün əhəmiyyətlidir. Cədvələr arası inteqrasiya, bir verilənlər bazasında yerləşən fərqli cədvələrin məlumatlarının bir-biri ilə əlaqələndirilməsi və birləşdirilməsi prosesidir. Bu, məlumatların doğru və düzgün şəkildə təşkil edilməsini, məlumatların bərabərliyini və əlaqəli məlumatların birləşdirilməsini təmin edir. Cədvələr arası inteqrasiya genelliklə əlaqələndirilmiş məlumatların cədvələr arasında birləşdirilməsi, cədvələr arasında əlaqələrin qurulması və məlumatların cədvələr arasında mübadiləsi ilə əlaqəlidir. Bu proses, verilənlər bazasında depolanmış məlumatları daha bərabər və əlaqəli şəkildə təşkil etmək üçün istifadə olunur. Cədvələr arası inteqrasiya üçün müxtəlif metodlar və texnologiyalar mövcuddur. Bu, cədvəllər arasında xüsusiyyət açarlarının təyin edilməsi, cədvəllər arasında əlaqələrin yaradılması üçün əlaqə açarlarının istifadəsi, cədvəllər arasında bir-biri ilə əlaqəli məlumatların birləşdirilməsi üçün JOIN əmrlərinin istifadəsi kimi prosedurların tətbiq edilməsini əhatə edir.

Cədvələr arası inteqrasiya, verilənlər bazasının təmiz, quruluşa uyğun və optimal şəkildə məlumatları təşkil etməsini təmin edir. Bu, məlumatların dəqiqliyini,

tamamını və məlumatların müxtəlif cədvəllər arasında birləşdirilməsini mümkün edir. Həmçinin, cədvələr arası inteqrasiya, verilənlər bazasının performansını və effektivliyini artırır və təkrarlanan məlumatları azaldır.

Bu məsələlərə diqqət yetirilməsi, məlumat bazasının effektiv işləməsini təmin edəcək və aerportun operasyonlarına yaxşı bir şəkildə cavab verəcək.

IV FƏSİL. Python proqramlaşdırma dilində yaratdıqım hava limanı məlumat bazsının kod hissəsi ilə tanış olaq. Bu kod hissəsini hissələrə bölünmüş şəkildə sizə təqdim edəcəyəm.

İlk öncə Python proqramlaşdırma dilində yazılmış kodumuzun kitabxansının tanıdıldığı ilk sətrinə baxaq:

```
import sqlite3
```

`import sqlite3` ifadəsi, Python prqramlaşdırma dilində yerləşdirilmiş olan SQLite modulunu proqramınıza daxil etmək üçün istifadə olunur. Bu modul, SQLite verilənlər bazasına əlaqə qurmaq, bazalar yaratmaq, cədvəllər yaratmaq, məlumatları əlavə etmək, güncəlləmək və silmək kimi əməliyyatları yerinə yetirməyə imkan verir.

SQLite, yüksək sürətlə işləyən, server tələb etməyən, özünəməxsus bir verilənlər bazası sistemidir. Bu, sadəcə bir fayl formatında saxlanan verilənlər bazası olduğu üçün tətbiqetmələrinizə sadəcə bir fayl əlavə etməklə əlaqə qurmağınızı təmin edir.

`sqlite3` modulunu yükləmək üçün əksər Python distributivlərində hazır gəlir və sistemdə mövcud olmalıdır. Əgər modul yüklənməmişdirsə, yükləmək üçün aşağıdakı pip komandasını işlədə bilərsiniz:

```
pip install sqlite3
```

Bu kod, Python paket idarəetmə sistemi olan `pip` vasitəsilə SQLite modulunu yükləyəcək. Bu şəkildə `sqlite3` modulunu uğurla yüklədikdən sonra, `import sqlite3` ifadəsini proqramınızın əvvəlində istifadə edə bilərsiniz və SQLite ilə əlaqəli funksiyaları və dəyişənləri istifadə edə bilərsiniz.

İndi isə kodumuzun növbəti hissəsi ilə tanış olaq:

```
1.1. # Verilənlər bazasına qoşulmaq və bağlantı yaratmaq
conn = sqlite3.connect('hava__limani.db')
cursor = conn.cursor()
```

Bu kod hissəsi, SQLite verilənlər bazasına qoşulmaq və bir kursor yaratmaq üçün istifadə olunur. Əməliyyatların addım-addım izahı aşağıdakı kimi olur:

1. **`sqlite3`** modulunu tanıdaq: İlk olaraq, `sqlite3` modulunu Python proqramına tanıdmaq lazımdır. Bu modul, SQLite verilənlər bazası əməliyyatlarını icra etmək üçün lazımi funksiyaları təmin edir.

2. Verilənlər bazasına qoşulun: **`sqlite3.connect('hava__limani.db')`** sətri ilə, **'hava__limani.db'** adlı SQLite verilənlər bazasına qoşulmaq mümkün olur.

`.db` uzantısı, SQLite verilənlər bazası faylının tipini göstərir. SQLite, yerli bir verilənlər bazası sistemi olaraq faylları `.db` uzantısı ilə saxlamağa imkan verir. SQLite verilənlər bazası, bir fayl şəklində saxlanılan özəl bir formatdır. Bu fayl, verilənlər bazasının cədvəllərini, sətrlərini, məlumat tiplərini və digər məlumatları saxlamaq üçün istifadə edilir. `.db` uzantılı fayl, SQLite verilənlər bazasının bir təmsilidir və içərisində verilənlər bazasının bütün məlumatlarını saxlayır. SQLite verilənlər bazası faylını yaratmaq və işlətmək üçün SQLite modulunu istifadə edə bilərsiniz. Bu modul, SQLite verilənlər bazasına bağlantı qurmaq, sorguları icra etmək, məlumatları əlavə etmək, güncəlləmək və silmək kimi əməliyyatları yerinə yetirməyə imkan verir. .db uzantılı fayllar, SQLite

verilənlər bazalarını saxlamaq və onlarla əməliyyat aparmaq üçün geniş şəkildə istifadə olunur.

Qoşulma yaratmaq üçün `connect()` funksiyasından istifadə edirik.

`connect()` SQLite modulunun bir metodu olaraq təklif edilir və SQLite verilənlər bazası ilə bağlantı yaratmaq üçün istifadə olunur. Bu funksiya aşağıdakı sintaksisdə istifadə olunur:

```
connect(database [, timeout, detect_types, isolation_level, check_same_thread, factory, cached_statements, uri])
```

Bu parametrlərə nəzərə alaq:

- **`database`:** Bağlanılacaq verilənlər bazasının fayl adı və ya fayl yolu. Əgər fayl mövcud deyilsə, yeni bir fayl yaradılacaq. `database` parameteri, `connect()` funksiyasına ötürülən bir argumentdir və verilənlər bazasının fayl adını və ya fayl yolunu təyin edir.

Verilənlər bazası, verilənlər və məlumatların saxlandığı bir fayl sistemidir. Fayl adı və ya fayl yolu vasitəsilə `connect()` funksiyası ilə verilənlər bazasına qoşulur və ya yeni bir verilənlər bazası yaradılır. Fayl adı, bir fayl adı olaraq təyin edilə bilər (məsələn, "mydatabase.db") və ya tam fayl yolunu təyin edə bilər (məsələn, "/path/to/database.db").

- **`timeout`:** İstifadəçiyə verilənlər bazasına bağlantı qurmağa çalışarkən gözləmək üçün maksimum vaxt müddəti (saniyə cinsində). Əgər müəyyən edilməz, 5 saniyə olacaq. `timeout` parameteri, `connect()` funksiyasının bir parametri olaraq istifadə edilir və verilənlər bazasına qoşulmaq üçün nəzərdə tutulan maksimum gözləmə vaxtını təyin edir. Bu parametr, əməliyyatların icrası üçün verilənlər bazası ilə əlaqə yaratmağa çalışan proseslərin gözləmə vaxtını tənzimləməyə imkan verir. `timeout` dəyəri, saniyələrlə təyin edilir və verilənlər bazası ilə əlaqə yaratmağa çalışan prosesin verilənlər bazası üzərində icra etdiyi

əməliyyatların tamamlanması üçün gözləmə vaxtını ifadə edir. Əgər proses verilənlər bazası ilə əlaqə yaratmaq üçün müddəti ərzində cavab ala bilməzsə, bir xəta tələb edəcək və `sqlite3.TimeoutError` xətası göstəriləcək. `timeout` dəyəri, əsasən 0 və ya mənfi olaraq təyin edilə bilər. 0 təyin edildikdə, heç bir gözləmə edilmir və proses yalnızca mövcud mənbələri yoxlamaq üçün cəhdlər edir. Mənfi dəyərlər isə bərabər olaraq sonsuz gözləməni təmsil edir, yəni proses cavab alıncaya qədər müvəqqəti olaraq bloklanacaq.

- **`detect_types`:** Dəyərləri çıxarış etmək üçün tip təyinatının aktiv olub-olmadığını göstərir. `detect_types` parametri, `connect()` funksiyasının bir parametri olaraq istifadə edilir və verilənlər bazasından gələn məlumatların otomatik olaraq tipini təyin etmək üçün istifadə olunan funksiyaların aktivləşdirilməsini tənzimləyir. Bu parametrin dəyəri əsasən boolean **(`True` və ya `False`)** olur. Əgər **`detect_types=True`** təyin edilsə, SQLite driver-i verilənlər bazasından gələn məlumatların tipini təyin etmək üçün avtomatik funksiyaları işlədəcəkdir. Bu, məlumatların dəqiqliyini və doğruluğunu artırmağa kömək edir. Əgər **`detect_types=False`** təyin edilsə, verilənlər bazasından gələn məlumatlar üçün tip təyini avtomatik olaraq baş verməyəcəkdir.

Əsas syntaxi:

```
sqlite3.connect(database, detect_types=...)
```

`detect_types` dəyəri boolean **(`True` və ya `False`)** olaraq təyin edilir. Məsələn, **`detect_types=True`** verilənlər bazasından gələn məlumatların tipinin avtomatik olaraq təyin edilməsini aktivləşdirir. Bu, SQLite driver-inin məlumatları daha doğru və uyğun tipdə işlətməsinə imkan verir. Əks halda, yəni **`detect_types=False`** təyin edildikdə, məlumatların tipi avtomatik olaraq təyin

edilməyəcək və istifadəçi məlumatların tipini əvvəlcədən bilməli və uyğun konversiyaları əldə etmək üçün əlavə tədbirlər görə bilməlidir.

- **`isolation_level`:** Əməliyyatların izolyasiya səviyyəsini təyin edir. Əgər müəyyən edilməz, bazası istifadə edən sənəd təyin edir. `isolation_level` parametri, `connect()` funksiyasının bir parametri olaraq istifadə edilir və işlənən əməliyyatların verilənlər bazasında hansı izolyasiya səviyyəsində icra ediləcəyini tənzimləyir. Bu parametrin dəyəri əsasən bir string olur və müxtəlif izolyasiya səviyyələrini təmsil edir. İzolyasiya səviyyələri, eyni anda aparılan bəzi əməliyyatların bir-birinə təsirinin necə idarə olunduğunu göstərir və verilənlər bazasının müxtəlif istifadəçilər və proseslər arasında hansı səviyyədə izolyasiya təmin etdiyini nəzərə alır.

Əsas izolyasiya səviyyələri aşağıdakılardır:

- **`None` (default):** Verilənlər bazası tərəfindən hansısa izolyasiya səviyyəsi təyin edilmir. İstifadəçi verilənlər bazasının tərəfindən qeyd edilmiş izolyasiya səviyyəsini istədiyi kimi seçə bilər.

- **`'DEFERRED'`:** Əməliyyatlar yalnız verilənlər bazasına yazıldıqda kilidlənməni başlatır. Kilidlənmə prosesi tam əməliyyat bitdikdə başlayır.

- **`'IMMEDIATE'`:** Əməliyyatlar verilənlər bazasına yazılanda kilidlənməni başlatır. Bu, digər əməliyyatların oxunmasını tələb edir.

- **`'EXCLUSIVE'`:** Verilənlər bazası yalnız bu proses tərəfindən istifadə oluna bilər. Digər proseslər bu verilənlər bazasına qoşula bilməz.

Əsas kodu:

```
sqlite3.connect(database, isolation_level=...)
```

`isolation_level` dəyəri string olaraq təyin edilir və müxtəlif izolyasiya səviyyələrini təmsil edir. Məsələn**, `isolation_level='DEFERRED'`** verilənlər bazasının izolyasiya səviyyəsinin əməliyyatlar yalnız verilənlər bazasına yazıldıqda kilidlənməni başlatmasını təmin edir. Bu, digər əməliyyatların oxunmasını tələb edir. Digər izolyasiya səviyyələrini isə IMMEDIATE` və `EXCLUSIVE` olaraq təyin edə bilərsiniz.

- `check_same_thread`: İstifadəçiyə yalnızca tək bir nişanda istifadəyə imkan verilir. `check_same_thread` parametri, SQLite verilənlər bazasının eyni iş parçacığından istifadə edilməsinin yoxlanılmasını tənzimləyir. Bu parametrin dəyəri boolean **(`True` və ya `False`)** ola bilər.

Əgər **`check_same_thread=True`** təyin olunarsa, verilənlər bazasına yalnız həmin yaradılma iş parçacığından əməliyyatlar icra edilə bilər. Bu deməkdir ki, fərqli iş parçacıqları arasında verilənlər bazası əməliyyatları üçün tələb olunan daxili müqavimətlər öhdəlik altında saxlanılır. Əgər digər iş parçacığı verilənlər bazasına əməliyyat tətbiq etmək istəsə, səhv alacaq.

Əks halda, yəni **`check_same_thread=False`** təyin olunarsa, hər hansı bir iş parçacığı verilənlər bazası ilə əlaqə yarada və əməliyyatlar yerinə yetirə bilər. Bu, verilənlər bazası əməliyyatlarının hər hansı bir iş parçacığından istifadə edilə biləcəyi anlamına gəlir. `check_same_thread` parametri SQLite verilənlər bazası modulunun güvənli vəziyyətini tənzimləmək üçün istifadə olunur. İş parçacıqlar arasında məlumat bölüşməsi riskini azaldır və təhlükəsizlik məhdudiyyətlərini qoruyur. Bu parametrin dəyərini məlumat bazasının istifadə şərtlərinə və tələblərinə uyğun şəkildə təyin etmək əhəmiyyətlidir.

- **`factory`:** Özəl bir bağlantı obyekti yaratmaq üçün bir adi verilənlər bazası nüsxəsini təyin edir. `factory` parametri, özəl bir verilənlər bazası obyektinin yaradılması üçün istifadə olunan istənilən bir Python sinfi təyin etmək üçün istifadə edilir. Bu parametrin dəyəri bir Python sinfi obyektidir. `sqlite3` modulu özəl olaraq `Connection` və `Cursor` sinflərini dəstəkləyir. Əgər `factory` parametri təyin edilməzsə, `connect()` funksiyası otomatik olaraq `Connection` sinfindən bir obyekt yaradır. Əgər fərqli bir verilənlər bazası sinfi təyin etmək istəyirsinizsə, özəl bir sinif yaratmaq və onu `factory` parametri kimi göndərmək kifayətdir. Bu, sizə verilənlər bazası əməliyyatlarının özəlliklərini genişləndirmək və modifiyə etmək imkanı verir.

Məsələn, xüsusi bir verilənlər bazası sinfi yaratmaq və `factory` parametri olaraq göndərmək üçün aşağıdakı addımaları izləyə bilərsiniz:

1. Özəl bir sinif yaratın, məsələn `MyDatabase`:

```
```
class MyDatabase(sqlite3.Connection):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # Özəl funksionalıqlar əlavə edin
        # ...
```

2. `connect()` funksiyasında `factory` parametrini özəl sinifinizi təyin edin:

```
```
conn = sqlite3.connect('database.db', factory=MyDatabase)
```
```

Bu halda `connect()` funksiyası `MyDatabase` sinfindən bir obyekt yaradacaq və siz özəl funksiyonalıqlarınızı əlavə edə bilərsiniz.
```

`factory` parametri verilənlər bazası əməliyyatlarını genişləndirmək və özəlləşdirmək üçün daha çox imkan yaratır. Bu yolla, verilənlər bazasının davranışını öz istəyinizə uyğun şəkildə tənzimləmək olar.

- **`cached_statements`:** İstifadəçiyə nəzərdə tutulmuş bəyanatların cədvələ bağlantısını necə idarə etmək istədiyini göstərir. `cached_statements` parametri, SQLite verilənlər bazasında hazırda yadda saxlanılan SQL ifadələrinin tənzimlənməsini və saxlanılmasını idarə etmək üçün istifadə olunur.

Əvvəlcədən hazırlanmış SQL ifadələri (məsələn, SELECT, INSERT, UPDATE, DELETE kimi əmrlər) verilənlər bazasında effektiv şəkildə icra edilmək üçün yadda saxlanılabilir. Bu, ifadələrin təkrarlanan icra zamanı təşkil olunmasını və kompilyasiya prosesindən keçməsini tələb etməz, buna görə də verilənlər bazasının performansını artırır.

`cached_statements` parametri iki dəyər qəbul edir: `True` və ya `False`. Varsayılan olaraq, parametrin dəyəri `False`-dur və hazırda yadda saxlanılan ifadələr yaratılmır. Əgər `cached_statements` dəyərini `True`-a təyin edərsiniz, `connect()` funksiyası yolu ilə açılan verilənlər bazasında hazırda yadda saxlanılan ifadələr yaratılacaq və saxlanılacaq. Bu, icra edilən ifadələrin daha sürətli olmasını və performansın artmasını təmin edir.

Məsələn, aşağıdakı kimi `cached_statements` dəyərini `True`-a təyin edə bilərsiniz

```
conn = sqlite3.connect('database.db', cached_statements=True)
```

Bu halda, verilənlər bazası ifadələrini yadda saxlamaq üçün yaddaşdan istifadə edəcək və təkrarlanan ifadələri təzədən kompilyasiya etmədən sürətlə icra edəcək. Bu, verilənlər bazasının performansını yüksəldəcəkdir.

- **`uri`:** Verilənlər bazasının üniversal resurs göstəricisini (URI) istifadə etmək üçün bir fayla keçirilir. `uri` parametri, SQLite veritabanına qoşularkən Uniform Resource Identifier (URI) ilə qoşulma üsulunu təyin etmək üçün istifadə olunur. URI, veritabanının yolu və parametrləri ilə birlikdə verilən bir stringdən ibarətdir.

SQLite URI, aşağıdakı sintaksisa əsaslanır:

```
sqlite:///path/to/database.db?param1=value1&param2=value2...
```

Bu sintaksisə əsasən, URI üç hissədən ibarət olur:

1. `sqlite:///` - SQLite veritabanına qoşulma protokolu.
2. `path/to/database.db` - Veritabanının diskdəki yolu və adı.
3. `param1=value1&param2=value2...`

İstəyə bağlı parametrlər, veritabanı qoşulmasını tənzimləmək üçün istifadə olunur. Məsələn, `timeout`, `detect_types`, `isolation_level` və s. kimi parametrlər URI-də təyin edilə bilər.

Nümunə olaraq, aşağıdakı kimi bir SQLite URI təyin edilə bilər:

```
conn = sqlite3.connect('sqlite:///path/to/database.db?timeout=10&detect_types=PARSE_DECLTYPES')
```

Bu halda, `path/to/database.db` faylına 10 saniyəlik bir timeout qoşulacaq və

veritabanında tipi tanıma üçün `PARSE_DECLTYPES` detekt etmə metodu istifadə ediləcəkdir. SQLite URI, veritabanına qoşulma zamanı əlavə parametrləri tənzimləmək üçün daha rahat və yüksək səviyyəli bir yolla sağlayır.

Bu `connect()` metodu çağırıldıqdan sonra, verilənlər bazasına olan bağlantı açılır və həmin bağlantı obyekti geri qaytarılır. Bu obyekt üzərində dərhal sorguları icra etmək, məlumatları əlavə etmək, güncəlləmək və silmək kimi əməliyyatlar aparılabilir.

Kodumuzun digər bir hissəsi ilə tanış olaq:

```
1.2. # Sərnişinlər cədvəlini yaratmaq
cursor.execute('''
    CREATE TABLE IF NOT EXISTS sernisinler (
        ID INTEGER PRIMARY KEY,
        Ad TEXT,
        Yas INTEGER,
        Cins TEXT,
        Telefon TEXT
    )
''')
```

Bu kod, SQLite məlumat bazsında "sernisinler" adlı bir cədvəlin yaradılması üçün istifadə olunur. Əməliyyat, **`cursor.execute()`** funksiyası vasitəsilə icra edilir. SQL dilində olan CREATE TABLE ifadəsi, yeni bir cədvəl yaratmaq üçün istifadə olunur.

Cədvəlin yaradılması prosesi zamanı istifadə olunan sütunlar aşağıdakılardır:

1. ID: INTEGER növündə olan bir sütun və əsas açarı (PRIMARY KEY) olaraq təyin edilmişdir. Bu sütun, hər bir sərnişin üçün unikal bir təyinat dəyəri saxlayır.

2. Ad: TEXT növündə olan bir sütun, sərnişinin adını təmsil edir.

3. Yas: INTEGER növündə olan bir sütun, sərnişinin yaşını saxlayır.

4. Cins: TEXT növündə olan bir sütun, sərnişinin cinsini təmsil edir.

5. Telefon: TEXT növündə olan bir sütun, sərnişinin telefon nömrəsini göstərir.

CREATE TABLE IF NOT EXISTS ifadəsi, "sernisinler" adlı cədvəlin artıq mövcud olub olmadığını yoxlayır. Əgər cədvəl daha əvvəl yaradılmışdırsa təkrar yaradılmaz və mövcud halı ilə istifadə olunmaya davam edilir. Bu, cədvəlin yalnız bir dəfə yaradılmasını təmin edir və mövcud məlumatların itirilməsinin qarşısını alır. Belə yaradılan "sernisinler" cədvəli, sərnişinlərə aid məlumatları saxlamaq üçün istifadə edilə bilər.

1.3 Sərnişin məlumatlarını əlavə etmək.

`sernisin_ekle` adlı bu funksiya, sərnişinlər cədvəlinə yeni bir sərnişinin məlumatlarını əlavə etmək üçün istifadə olunur. Funksiya aşağıdakı kod bloku vasitəsilə icra olunur:

```
def sernisin_ekle(ad, yas, cins, telefon):
    cursor.execute('''
        INSERT INTO sernisinler (Ad, Yas, Cins, Telefon)
        VALUES (?, ?, ?, ?)
    ''', (ad, yas, cins, telefon))
    conn.commit()
    #print("Sərnişin məlumatları əlavə edildi.")
```

`def` açar sözü Python proqramlarında funksiya təyin etmək üçün istifadə olunur. `def` ifadəsindən sonra funksiyanın adı gəlir, ardından parametrlər mövcuddursa onlar qəbul edilir. Funksiyanın işləmə meydana gətirməsi üçün lazım olan kod bloku, girintilmiş şəkildə altında yazılır. Bu nümunədə `sernisin_ekle` adlı bir funksiya təyin edilib. İstifadəçidən `ad`, `yas`, `cins` və `telefon` parametrləri əldə edilir. Sonra `cursor.execute()` ilə verilənlər bazasına SQL əməliyyatı göndərilir və dəyişikliklər `conn.commit()` vasitəsilə təsdiqlənir.

Funksiyalar, proqramda bəzi əməliyyatları təkrarlayan və bir araya gətirən modular strukturlar yaratmaq üçün istifadə olunur. Onlar kodun oxunaqlığını artırır və funksiyanın bir neçə yerdə istifadə edilməsi halında dəyişikliklərin sadəcə bir dəfə edilməsinə imkan verir.

Funksiya dörd ədəd parametr qəbul edir: `ad`, `yas`, `cins`, `telefon`. Bu parametrlər vasitəsilə sərnişinin adı, yaşı, cinsi və telefon nömrəsi əlavə edilir.

Əməliyyat SQL dilində olan INSERT INTO ifadəsi ilə həyata keçirilir.

`INSERT INTO` ifadəsi, SQLite verilənlər bazasında yeni sətir (qeyd) əlavə etmək üçün istifadə olunur.

`INSERT INTO` ifadəsinin sintaksisi aşağıdakı kimi olur:

```sql
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Burada:

- `table_name`: Yeni sətir əlavə ediləcək cədvəlin adı.

- `(column1, column2, column3, ...)`: Əlavə ediləcək sütunların adları.

- `(value1, value2, value3, ...)`: Sütunlara əlavə ediləcək dəyərlər.

Məsələn, əvvəlki kod nümunəsində, `sernisinler` cədvəlinə yeni sərnişin məlumatları əlavə etmək üçün aşağıdakı `INSERT INTO` ifadəsi istifadə edilir:

```
INSERT INTO sernisinler (Ad, Yas, Cins, Telefon)
VALUES (?, ?, ?, ?);
```

Bu ifadədə `sernisinler` cədvəlinin adı verilib və sütunlar (`Ad`, `Yas`, `Cins`, `Telefon`) təyin edilir. `VALUES` hissəsində isə dəyişən dəyərlər (`?`) istifadə edilir. Bu dəyərlər daha sonra `cursor.execute()` funksiyasında təyin edilən tuple dəyərləri ilə əvəzlənir. Buna görə də kodun əlavə etmək istədiyimiz konkret sərnişin məlumatlarına uyğun olaraq dəyişdirilməsi vacibdir.

`cursor.execute()` metodu, SQLite verilənlər bazasında SQL əmrlərini icra etmək üçün istifadə olunan bir metoddur. Bu metod, verilənlər bazası ilə bağlantı qurulan `cursor` obyektində mövcuddur.

`cursor.execute()` metodunun sintaksisi aşağıdakı kimi olur:

```
cursor.execute(sql_query, parameters)
```

Burada:

- **`sql_query`:** SQL əmrini təmsil edən bir stringdir. Bu əmrdə SELECT, INSERT, UPDATE, DELETE kimi əmrlər olabilir. `sql_query`, `cursor.execute()` metodunda əmr kimi icra ediləcək SQL ifadəsini təmsil edən bir stringdir. Bu ifadə, verilənlər bazasında icra olunacaq əmri müəyyən edir.

SQL ifadələri, verilənlər bazası əmrlərinin formasına əsasən yazılır və müxtəlif məqsədlərlə istifadə olunur. SQL ifadələri SELECT, INSERT, UPDATE, DELETE kimi əmrlərə uyğun gələn fərqli funksiyonalıqları icra edə bilər.

Məsələn, aşağıdakı SQL ifadələrindən bir neçəsinin nümunələrini göstərək:

- SELECT ifadəsi: `SELECT * FROM sernisinler`

Bu ifadə, "sernisinler" cədvəlindəki bütün sətirləri seçir.

- INSERT ifadəsi: `INSERT INTO sernisinler (Ad, Yas, Cins, Telefon) VALUES ('John Doe', 25, 'Male', '555-1234')`

Bu ifadə, "sernisinler" cədvəlinə yeni bir sətir əlavə edir və müvafiq sütunlara dəyərləri təyin edir.

- UPDATE ifadəsi: `UPDATE sernisinler SET Yas = 30 WHERE Ad = 'John Doe'`

Bu ifadə, "sernisinler" cədvəlində "Ad" sütununda "John Doe" adı ilə eşləşən sətirlərin "Yas" sütununu 30 ilə yeniləyir.

- DELETE ifadəsi: `DELETE FROM sernisinler WHERE Yas > 40`

Bu ifadə, "sernisinler" cədvəlində "Yas" sütunu 40-dan böyük olan sətirləri silir.

SQL ifadələri, verilənlər bazasında əmrlərin icrası və məlumatların idarə olunması üçün əsas alınır. Mövcud verilənləri seçmək, əlavə etmək, yeniləmək və silmək üçün uyğun ifadələr yazılır.

- `parameters` (isteğe bağlı): SQL əmri içində istifadə ediləcək parametrlərdir.

`parameters`, `cursor.execute()` metodunda SQL ifadəsinə daxil edilən dəyərlərin yerləşəcəyi yerlərə verilən məlumatlardır. Bu məlumatlar SQL ifadəsində yerləşən yer tutucuların (`?` simvolu ilə göstərilən) dəyərləri təmsil edir.

SQL ifadəsində dəyişəcək dəyərləri dinamik olaraq təyin etmək və verilənlər bazasına daxil edilməsi üçün parametrizasiya (parameterization) tətbiq etmək vacibdir. Bu, SQL ifadələrinin daha təhlükəsiz, daha təmiz və daha effektiv olmasına kömək edir.

`cursor.execute()` metodunda `parameters` parameteri ilə göstərilən dəyərlər SQL ifadəsinin yer tutucularına səs-küy vermək üçün istifadə olunur. Bu dəyərlər əsasən tuple (demək olar ki, sıralıq) şəklində verilir.

Məsələn, aşağıdakı misalda `cursor.execute()` metodu kullanılarak bir INSERT SQL ifadəsi icra edilir və dəyərlər `parameters` parametrindən alınır:

```
name = "John Doe"
age = 25
gender = "Male"
cursor.execute("INSERT INTO sernisinler (Ad, Yas, Cins) VALUES (?, ?, ?)", (name, age, gender))
```

Burada `Ad`, `Yas` və `Cins` sütunlarına uyğun olaraq `name`, `age` və `gender` dəyərləri SQL ifadəsindən ayrıca verilmədən `parameters` tuple-ləri vasitəsilə əlavə edilir.

`parameters` tuple-i, SQL ifadəsindəki yer tutucuların sırası ilə uyğun olmalıdır. Bu, dəyərlərin düzgün sütunlara yerləşdirilməsinə əmin olmağa kömək edir.

Əgər SQL əmrində parametrlər varsa, onların dəyərləri bu parametrlərdə təyin edilir.

Məsələn, aşağıdakı kod nümunəsində `cursor.execute()` metodu istifadə olunur:

```
cursor.execute("SELECT * FROM sernisinler")
```

Bu əmr SQL əmrləri üçün ən çox istifadə olunan `SELECT * FROM table_name` əmri ilə bütün sətirləri seçir. SQL əmri string şəklində `cursor.execute()` metoduna təqdim edilir. Əmrlərə parametrlər daxil etmək istədiyimiz zaman isə aşağıdakı kimi `parameters` argumentini əlavə edə bilərik:

```
cursor.execute(
    "INSERT INTO sernisinler (Ad, Yas, Cins, Telefon) VALUES (?, ?, ?, ?)", (ad, yas, cins, telefon))
```

Bu nümunədə, `INSERT INTO` SQL əmri daxil edilir və dəyişən dəyərlər (`ad`, `yas`, `cins`, `telefon`) əlavə edilmək üçün `parameters` argumentində tuple olaraq təqdim edilir.Əmrlər icra olunduqdan sonra, nəticələrə və ya əmrlərin təsdiqlənməsinə əsasən fərqli əmrlərdən istifadə olunacaq. Nəticələri alarkən `fetchone()`, `fetchall()` və s. kimi metodlardan istifadə edə bilərik. Təsdiqləmə

edilmiş əmrlərdən sonra `conn.commit()` metodu çağırılaraq dəyişikliklər verilənlər bazasına qeyd edilir.

Bu metodun SQLite verilənlər bazasında əmrlərin icrası və nəticələrin idarə olunması üçün əsaslı bir rolu var.

`cursor.execute()` metodu, SQL ifadəsini verilənlər bazasına göndərir. Ədədləri və ya mətni SQL ifadəsinin içindəki uyğun yerlərə (`?` simgəni) yerləşdiririk. Bu, SQL ifadəsinin təhlil prosesində dəyərlərin doğru formada yerləşdirilməsinə imkan verir və SQL enjeksiya hücumlarından qoruyur.

Sonra `conn.commit()` funksiyası çağırılır. Bu, əməliyyatların verilənlər bazasına təsdiq edilməsini və dəyişikliklərin saxlanılmasını təmin edir.

Əlavə olaraq, funksiyada bir sətir yoxdur qeyd olunmuşdur (`#print("Sərnişin məlumatları əlavə edildi.")`). Əgər sərnişin məlumatlarının əlavə edilmə prosesinin nəticələrini görmək istəyirsinizsə, bu sətiri əlavə edərək nəticəni ekrana çıxara bilərsiniz.

1.4 Sərnişin məlumatlarını güncəlləmək.

```
# Sərnişin məlumatlarını güncəlləmək
def sernisin_guncelle(id, ad, yas, cins, telefon):
    cursor.execute('''
        UPDATE sernisinler
        SET Ad=?, Yas=?, Cins=?, Telefon=?
        WHERE ID=?
    ''', (ad, yas, cins, telefon, id))
    conn.commit()
    print("Sərnişin məlumatları güncəlləndi.")
```

`sernisin_guncelle()` adlı bu funksiya, verilənlər bazasındakı "sernisinler" cədvəlindəki bir sərnişinin məlumatlarını güncəlləmək üçün istifadə olunur.

Funksiya `cursor.execute()` metodunu kullanarak bir UPDATE SQL ifadəsi icra edir. SQL ifadəsi, `sernisinler` cədvəlindəki sərnişin məlumatlarını yeniləmək üçün nəzərdə tutulmuşdur. Yer tutucular (`?` simvolu ilə göstərilən) vasitəsilə dəyərlər SQL ifadəsinə daxil edilir. Dəyişdirilən sütunlar `SET` hissəsində göstərilir və verilən `id` dəyəri ilə müəyyən sərnişin təyin olunur.

Sonra, `conn.commit()` əmrindən istifadə edilir, böyük həcmli dəyişiklikləri verilənlər bazasında etibarlı şəkildə təsdiq etmək üçün. Nihayət, "Sərnişin məlumatları güncəlləndi." mesajı çıxışa yazdırılır.

Məsələn, aşağıdakı koda görə, `sernisin_guncelle()` funksiyası əmr verildikdə `id` dəyəri ilə müəyyən sərnişinin məlumatlarını yeniləyəcək:

```
sernisin_guncelle(1, 'Fatime Hesenli', 25, 'Qadın', '+994516780909')
```

Bu kod, `id` dəyəri 1 olan sərnişinin adını "Fatime Hesenli", yaşını 25, cinsini

"Qadın" və telefon nömrəsini "+994516780909" olaraq dəyişdirir. Əgər verilənlər bazasında bu `id`-yə uyğun bir sərnişin mövcuddursa, məlumatlar yenilənir və "Sərnişin məlumatları güncəlləndi." mesajı çıxışa yazdırılır.

1.5. Sərnişinləri silmək.

```
def sernisin_sil(id):
    cursor.execute('''
        DELETE FROM sernisinler
        WHERE ID=?
    ''', (id,))
    conn.commit()
    print("Sərnişin silindi.")
```

`sernisin_sil()` adlı bu funksiya, verilənlər bazasında "sernisinler" cədvəlində bir sərnişini silmək üçün istifadə olunur.

Funksiya `cursor.execute()` metodu vasitəsilə DELETE SQL ifadəsini icra edir. SQL ifadəsi, `sernisinler` cədvəlindəki sərnişinləri silmək üçün nəzərdə tutulmuşdur. Yer tutucusu (`?` simvolu ilə göstərilən) vasitəsilə `id` dəyəri SQL ifadəsinə daxil edilir. Bu, silinəcək sərnişinin müəyyən edilməsi üçün istifadə edilir

Daha sonra, `conn.commit()` əmrindən istifadə edilir, böyük həcmli dəyişiklikləri verilənlər bazasında etibarlı şəkildə təsdiq etmək üçün. Nihayət, "Sərnişin silindi." mesajı çıxışa yazdırılır.

Məsələn, aşağıdakı koda görə, `sernisin_sil()` funksiyası `id` dəyəri ilə müəyyən sərnişini siləcək:

```
sernisin_sil(1)
```

Bu kod, `id` dəyəri 1 olan sərnişiniyi silir. Əgər verilənlər bazasında bu `id`-yə uyğun bir sərnişin mövcuddursa, o silinir və "Sərnişin silindi." mesajı çıxışa yazdırılır.

1.6. Sərnişin məlumatlarını göstərmək.

```
def sernisinleri_goster():
    # Sərnişin məlumatlarını seçmək üçün SQL ifadəsini icra et
    cursor.execute("SELECT * FROM sernisinler")
    # SQL ifadəsindən bütün sərnişin məlumatlarını əldə et
    sernisinler = cursor.fetchall()
    # Hər bir sərnişin üçün bir dövr et
    for sernisin in sernisinler:
        # Sərnişin məlumatlarını çıxışa yazdır
        print(f"ID: {sernisin[0]}, Ad: {sernisin[1]}, Yaş: {sernisin[2]}, Cins: {sernisin[3]}, Telefon: {sernisin[4]}")
```

`sernisinleri_goster()` adlı bu funksiya, verilənlər bazasında "sernisinler" cədvəlindəki bütün sərnişin məlumatlarını göstərmək üçün istifadə olunur.

Funksiya `cursor.execute()` metodu vasitəsilə SELECT SQL ifadəsini icra edir. SQL ifadəsi, bütün sütunları (yıldız işarəsi `*`) və "sernisinler" cədvəlini seçir.

Sonra `cursor.fetchall()` metodu ilə bütün sərnişin məlumatları alınır. Bu metod, icra edilən SQL sorgusunun bütün nəticələrini bir siyahı şəklində qaytarır.

Dövr vasitəsilə hər bir sərnişin üçün bir `sernisin` obyekti yaradılır. Bu obyektin indeksləri ilə sərnişin məlumatlarına (`ID`, `Ad`, `Yaş`, `Cins`, `Telefon`) müraciət edilir. Bu məlumatlar çıxışa yazdırılır.

Aşağıda funksiyanın parçalara bölünmüş variantı ilə bağlı ətraflı məlumatlar verilmişdir:

```
def sernisinleri_goster():
    # Sərnişin məlumatlarını seçmək üçün SQL ifadəsini icra et
    cursor.execute("SELECT * FROM sernisinler")
    # SQL ifadəsindən bütün sərnişin məlumatlarını əldə et
    sernisinler = cursor.fetchall()
    # Hər bir sərnişin üçün bir dövr et
    for sernisin in sernisinler:
        # Sərnişin məlumatlarını çıxışa yazdır
        print(f"ID: {sernisin[0]}, Ad: {sernisin[1]}, Yaş: {sernisin[2]}, Cins: {sernisin[3]}, Telefon: {sernisin[4]}")
```

Bu kod, "sernisinler" cədvəlindəki bütün sərnişin məlumatlarını əldə edir və hər

bir sərnişinin məlumatlarını çıxışa yazdırır. Bu funksiyanı çağırdıqda bütün sərnişin məlumatları sıralanır.

1.7. Sərnişin əməliyyatlarını sınamaq.

```
sernisin_ekle('Fatime Hesenli', 24, 'Qadın', '+994516780909')
sernisin_ekle('Ballı Memmedova', 45, 'Qadın', "+994516760909")
sernisin_ekle('Nermin Novruzova', 28,'Qadın', "+994516770909")
sernisin_ekle('Elmar Aliev', 27, 'Kişi', '+994516750909')
sernisin_ekle('Samir Ceferli', 75, 'Kişi', "+994516740909")
sernisin_ekle('Rehile Yolçuyeva', 34, 'Qadın', "+994516730909")
sernisin_ekle('Aysu İbrahimova ', 41, 'Qadın', '+994516720909')
sernisin_ekle('Jale İsgenderova', 32, 'Qadın', "+994516710909")
sernisin_ekle('Natali Adekugbe', 35, 'Qadın', "+994516700909")
sernisin_ekle('Jane Smith', 25, 'Qadın', '+994516690909')
sernisin_ekle('Mauro İcardi', 29, 'Kişi', "+994516680909")
sernisin_ekle('Lucas Toreira', 26, 'Kişi', "+994516670909")
```

Yazdığınız kod, "sernisin_ekle" fonksiyonunu çağırarak sərnişin məlumatlarını veritabanına əlavə etmək üçün istifadə edir. Aşağıda verdiyimiz sərnişin məlumatları ilə əməliyyatları sınaqdan keçirmək üçün bu kodu izah edəcəyik:

```
sernisin_ekle('Fatime Hesenli', 24, 'Qadın', '+994516780909')
sernisin_ekle('Ballı Memmedova', 45, 'Qadın', "+994516760909")
sernisin_ekle('Nermin Novruzova', 28,'Qadın', "+994516770909")
sernisin_ekle('Elmar Aliev', 27, 'Kişi', '+994516750909')
sernisin_ekle('Samir Ceferli', 75, 'Kişi', "+994516740909")
sernisin_ekle('Rehile Yolçuyeva', 34, 'Qadın', "+994516730909")
sernisin_ekle('Aysu İbrahimova ', 41, 'Qadın', '+994516720909')
sernisin_ekle('Jale İsgenderova', 32, 'Qadın', "+994516710909")
sernisin_ekle('Natali Adekugbe', 35, 'Qadın', "+994516700909")
sernisin_ekle('Jane Smith', 25, 'Qadın', '+994516690909')
sernisin_ekle('Mauro İcardi', 29, 'Kişi', "+994516680909")
sernisin_ekle('Lucas Toreira', 26, 'Kişi', "+994516670909")
```

Bu kod, "sernisin_ekle" fonksiyonunu çağırarak 12 fərqli sərnişin məlumatını

veritabanına əlavə edir. Hər bir `sernisin_ekle` çağrısı, sərnişin məlumatlarını təyin edir və bu məlumatları `sernisin_ekle` funksiyasına ötürür.

Əmrlər, "sernisin_ekle" funksiyasında təyin olunan məlumatları `cursor.execute()` metodu ilə INSERT INTO SQL ifadəsinə əlavə edir. `sernisin_ekle` funksiyası veritabanı ilə bağlantı yaratmaq və dəyişiklikləri təsdiqləmək üçün `conn.commit()` əmri istifadə edir.

Bu kod nəticəsində məlumat bazasında 12 sərnişin məlumatı yaradılacaq. Bu sərnişinlərin adı, yaşları, cinsləri və telefon nömrələri verilmişdir.

1.8. Hava limanı məlumat bazasından məlumatların silinməsi.

```
#sernisin_guncelle(1,'Gündüz Bayramov', 25, 'Kişi', "+994516770909")
#sernisin_sil(2)
sernisinleri_goster()
```

Yazdığınız kodda, öncə `sernisin_guncelle` və `sernisin_sil` funksiyalarını çağırmışıq və ardından `sernisinleri_goster` funksiyasını çağırmışıq. Gəlin indi bu kod parçalarının açıqlamalarına baxaq:

```
sernisin_guncelle(1,'Gündüz Bayramov', 25, 'Male', "+994516770909")
```

Bu kod, `sernisin_guncelle` funksiyalarını çağıraraq ID'si 1 olan sərnişin məlumatını güncəllər. Yeni məlumatlar `Ad='Gündüz Bayramov'`, `Yas=25`, `Cins='Male'` və `Telefon='+994516770909'` olaraq təyin edilmişdir.

```
sernisin_sil(2)
```

Bu kod, `sernisin_sil` funksiyalarını çağıraraq ID'si 2 olan sərnişini silər. Məlumat bazasında ID'si 2 olan sərnişin məlumatı silinir.

```
sernisinleri_goster()
```

Bu kod, `sernisinleri_goster` funksiyalarını çağıraraq məlumat bazsındaki sərnişin məlumatlarını göstərir. SELECT sorğusu vasitəsilə bütün sərnişin məlumatları alınır və ekrana çıxarılır. İstifadə etdiyiniz kodlar vasitəsilə sərnişin məlumatlarını güncəlləyə bilə, silə bilərsiniz və məlumat bazsındaki bütün sərnişin məlumatlarını göstərə bilərsiniz. Bu, sərnişin əməliyyatlarını idarə etmək üçün əlavə funksiyalara misal olaraq verilmişdir.

1.9. Bağlantını bağlamaq.

```
conn.close()
```

`conn.close()` əmri, SQLite verilənlər bazası ilə olan əlaqəni bağlamaq üçün istifadə olunur. Bu əmr, açılan verilənlər bazası bağlantısını bağlayacaq və baza ilə olan bütün əməliyyatlar sonlandırılacaq.

Bağlantı bağlandıqdan sonra artıq verilənlər bazası ilə əməliyyatlar edə bilməzsiz. Əgər daha sonra yenidən əməliyyatlar etmək istəyirsinizsə, yenidən `sqlite3.connect()` əmrini işlədərək yeni bir bağlantı yaratmalısınız.

Hava limanı məlumat bazasının yaratmaq üçün bizə lazım olan proqramımızın kodu ilə hissə-hissə tanış olduq. İndi kodun ümumi təsvirinə baxaq:

```
import sqlite3

# Verilənlər bazasına qoşulmaq və bağlantı yaratmaq
conn = sqlite3.connect('haaaaava_limani.db')
cursor = conn.cursor()

# Sərnişinlər cədvəlini yaratmaq
cursor.execute('''
    CREATE TABLE IF NOT EXISTS sernisinler (
        ID INTEGER PRIMARY KEY,
        Ad TEXT,
        Yas INTEGER,
        Cins TEXT,
        Telefon TEXT
    )
''')

# Sərnişin məlumatlarını əlavə etmək
def sernisin_ekle(ad, yas, cins, telefon):
    cursor.execute('''
        INSERT INTO sernisinler (Ad, Yas, Cins, Telefon)
        VALUES (?, ?, ?, ?)
    ''', (ad, yas, cins, telefon))
    conn.commit()
    #print("Sərnişin məlumatları əlavə edildi.")

# Sərnişin məlumatlarını güncəlləmək
def sernisin_guncelle(id, ad, yas, cins, telefon):
    cursor.execute('''
        UPDATE sernisinler
        SET Ad=?, Yas=?, Cins=?, Telefon=?
        WHERE ID=?
    ''', (ad, yas, cins, telefon, id))
    conn.commit()
    print("Sərnişin məlumatları güncəlləndi.")

# Sərnişinləri silmək
def sernisin_sil(id):
    cursor.execute('''
        DELETE FROM sernisinler
        WHERE ID=?
    ''', (id,))
    conn.commit()
    print("Sərnişin silindi.")
```

```python
# Sərnişin məlumatlarını göstərmək
def sernisinleri_goster():
    cursor.execute("SELECT * FROM sernisinler")
    sernisinler = cursor.fetchall()
    for sernisin in sernisinler:
        print(f"ID: {sernisin[0]}, Ad: {sernisin[1]}, Yaş: {sernisin[2]}, Cins: {sernisin[3]}, Telefon: {sernisin[4]}")

# Sərnişin əməliyyatlarını sınamaq
sernisin_ekle('Fatime Hesenli', 24, 'Qadın', '+994516780909')
sernisin_ekle('Ballı Memmedova', 45, 'Qadın', "+994516760909")
sernisin_ekle('Nermin Novruzova', 28,'Qadın', "+994516770909")
sernisin_ekle('Elmar Aliev', 27, 'Kişi', '+994516750909')
sernisin_ekle('Samir Ceferli', 75, 'Kişi', "+994516740909")
sernisin_ekle('Rehile Yolçuyeva', 34, 'Qadın', "+994516730909")
sernisin_ekle('Aysu İbrahimova ', 41, 'Qadın', '+994516720909')
sernisin_ekle('Jale İsgenderova', 32, 'Qadın', "+994516710909")
sernisin_ekle('Natali Adekugbe', 35, 'Qadın', "+994516700909")
sernisin_ekle('Jane Smith', 25, 'Qadın', '+994516690909')
sernisin_ekle('Mauro İcardi', 29, 'Kişi', "+994516680909")
sernisin_ekle('Lucas Toreira', 26, 'Kişi', "+994516670909")
#sernisin_guncelle(1,'Gündüz Bayramov', 25, 'Male',"++994516770909")
#sernisin_sil(2)
sernisinleri_goster()

# Bağlantını bağlamaq
conn.close()
```

V FƏSİL. Yaradılan hava limanı məlumat bazasının qiymətləndirilməsi:

Yaradılan məlumat bazası, SQLite verilənlər bazası sistemində bəzi qiymətləndirmələrə əsaslanaraq qiymətləndirilə bilir:

1. Funksional vərəqəlilik: Məlumat bazası, tələb olunan funksional dəstəyi təmin edir. Bu, tabloların yaradılması, sərnişinlərin məlumatlarının əlavə edilməsi, güncəllənməsi və silinməsi, məlumatların sorgulanması və digər əməliyyatların icrasını içərir

2. Performans: Məlumat bazasının performansı, sərnişinlərin məlumatlarının effektiv şəkildə əlavə edilməsi, güncəllənməsi, silinməsi və sorgulanması ilə bağlıdır. İndeks və uyğun indeksləmə tətbiq edilməsi, performansı artırmaq üçün əlavə tədbirlər ola bilər.

3. Məlumat təhlükəsizliyi: Məlumat bazasının təhlükəsizliyi, verilənlərin korunmasını və istifadəçilərin məlumatlara sadece uyğun icazə ilə daxil olmasını təmin edən bir sıra tədbirlərdən ibarətdir. Bu, məlumatların şifrələnməsi, istifadəçi hüquqlarının idarə edilməsi və potensial təhlükələrə qarşı qorunma tədbirlərini əhatə edir.

4. Təhlükəsizlik: Məlumat bazasının təhlükəsizliyi, mövcud təhlükəsizlik açıqlarının minimuma endirilməsi, istifadəçi hüquqlarının və məlumatların uyğun idarə edilməsi, güncəl məlumat bazası versiyasının istifadə edilməsi və təhlükəsizlik tədbirlərinin müvafiq olaraq tətbiq edilməsi ilə əlaqəlidir.Bu kriteriyaları qiymətləndirərək məlumat bazasının effektivlik, performans, təhlükəsizlik və funksional vərəqəlilik baxımından qiymətləndirmək

mümkündür. Bu qiymətləndirmə, məlumat bazasının tələblərə cavab verən və uyğun səviyyədə işləyən bir sistem olduğunu müəyyən etmək üçün vacibdir.

1.Yaradılan məlumat bazasının qiymətləndirilməsi üçün aşağıdakı testlər aparıla bilər:

1.1. Tabloların yaradılması: Yaradılan sərnişinlər cədvəlinin mövcud olub-olmaması yoxlanılmalıdır. Yoxlamaq üçün SQL əmrini icra edib, cədvəlin mövcudluğunu yoxlaya bilərsiniz.

1.2. Sərnişin məlumatlarının əlavə edilməsi: `sernisin_ekle` funksiyası ilə sərnişin məlumatları əlavə edilməlidir. Əlavə edilən məlumatların cədvəldə düzgün şəkildə saxlanılıb-saxlanılmadığı yoxlanılmalıdır.

1.3. Sərnişin məlumatlarının güncəllənməsi: `sernisin_guncelle` funksiyası ilə sərnişin məlumatları güncəllənməlidir. Güncəllənən məlumatların doğru bir şəkildə cədvəldə dəyişdirildiyi yoxlanılmalıdır.

1.4. Sərnişinlərin silinməsi: `sernisin_sil` funksiyası ilə sərnişinlər silinməlidir. Silinən sərnişinlərin artıq cədvəldə olmadığının yoxlanılması lazımdır.

1.5. Sərnişinlərin göstərilməsi: `sernisinleri_goster` funksiyası ilə cədvəldəki sərnişinlər göstərilməlidir. Göstərilən məlumatların sərnişinlər bazasındakı məlumatlarla uyğun olduğu yoxlanılmalıdır.

Bu testlər vasitəsilə məlumat bazasının əməliyyatlarının düzgün işlədiyini və məlumatların cədvəldə düzgün şəkildə saxlandığını yoxlamaq mümkündür. Hər bir əməliyyatın uğurlu şəkildə tamamlanması və cədvəldə gözlənilən dəyişikliklərin gerçəkləşdirilməsi, məlumat bazasının effektiv və düzgün işlədiyini göstərir.

V FƏSİL. Nəticə və mülahizə:

1. Hava limanı məlumat bazasının yaradılması nəticəsində aşağıdakılar əldə edilə bilir:

1. Hava limanı məlumatları: Məlumat bazası hava limanları haqqında məlumatların saxlanılmasını təmin edir. Hava limanlarının adı, ünvanı, koordinatları, terminal sayı, uçuşların siyahısı və digər məlumatlar bu məlumat bazasında qeyd edilə bilər.

2. Uçuş məlumatları: Hava limanı məlumat bazası, uçuşların haqqında ətraflı məlumatların saxlanılmasına imkan verir. Bu məlumatlar uçuşun növü, uçuş saatı, hava yolu şirkətinin adı, uçuş nömrəsi, uçuş istiqaməti və s. kimi məlumatları əhatə edə bilər.

3. Sərnişin məlumatları: Məlumat bazasında sərnişinlərin adı, soyadı, yaşı, cinsi, əlaqə nömrəsi və digər şəxsi məlumatlar da saxlanıla bilər. Bu məlumatlar, uçuşlarla bağlı rezervasiya məlumatlarının saxlanılması və sərnişinlərin təhlükəsizlik prosedurlarından keçməsi üçün işləyə bilər.

4. Baqaj məlumatları: Hava limanı məlumat bazası, uçuşlarla əlaqədar olan bagaj məlumatlarının saxlanılmasına imkan verir. Bu, bagajın növü, həcmi, çəkisi, bagajın sahibi və digər məlumatları əhatə edə bilər.

2. Yaradılan hava limanı məlumat bazası, çeşidli sərnişinlərə, hava limanlarına və uçuş məlumatlarına dair məlumatların saxlanılmasını və idarə edilməsini təmin edir. Bu, hava limanı operatorları, hava yolu şirkətləri və sərnişinlər üçün bir sıra faydalar yaradır:

1. Məlumatların effektiv idarə olunması: Hava limanı məlumat bazası sayəsində sərnişinlər, hava limanları və uçuş məlumatları ilə əlaqədar məlumatlar bir yerdə

toplanır. Bu, məlumatların daha effektiv şəkildə idarə olunmasını və tənzimlənməsini təmin edir

2. Rezervasiya və bilet idarəsi: Sərnişinlərin uçuş rezervasiya və bilet məlumatları hava limanı məlumat bazasında saxlanılır. Bu, sərnişinlərin uçuşları üzrə rezervasiya məlumatlarının effektiv şəkildə idarə olunmasını və bilet proseslərinin təhlükəsiz şəkildə yerinə yetirilməsini təmin edir.

3. Uçuş məlumatlarının tənzimlənməsi: Hava limanı məlumat bazası, uçuş məlumatlarının saxlanılmasına və idarə olunmasına imkan verir. Bu, uçuşların saatları, növləri, hava yolu şirkətləri və digər məlumatların effektiv şəkildə tənzimlənməsini və uçuşların idarə olunmasını təmin edir

4. Sərnişin məlumatlarının təhlükəsizliyi: Məlumat bazası, sərnişinlərə aid şəxsi məlumatların təhlükəsiz şəkildə saxlanılmasını təmin edir. Bu, sərnişinlərə aid məlumatların izlənməsi, təhlükəsizlik prosedurlarının icrası və identifikasiya proseslərinin effektiv şəkildə yerinə yetirilməsini təmin edir.

Yaradılan hava limanı məlumat bazası, hava limanı proseslərini daha təhlükəsiz, effektiv və səmərəli şəkildə idarə etmək üçün bir vasitədir. Bu, sərnişinlərə daha yaxşı xidmət göstərilməsinə imkan verir, hava limanı operatorlarının və hava yolu şirkətlərinin iş proseslərini avtomatlaşdırır və geniş məlumat analitikasına əsas təşkil edir. Bu cür məlumat bazaları, hava limanlarının effektivliyini artırmaq və səyahət proseslərini daha yaxşı idarə etmək üçün əhəmiyyətli bir vasitədir.

İstifadə olunmuş ədəbiyyatların siyahısı

1. Database Systems: Design, Implementation, and Management by Carlos Coronel, Steven Morris, and Peter Rob

2. SQL in 10 Minutes, Sams Teach Yourself by Ben Forta

3. SQLite Database System: Design and Implementation by Sibsankar Haldar and Raymond J. McCall

4. Practical SQL: A Beginner's Guide to Storytelling with Data by Anthony DeBarros

5. Learning SQL: Generate, Manipulate, and Retrieve Data by Alan Beaulieu

6. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design by Michael J. Hernandez

7. "Python SQLite3: Database Programming" by Jay SivaLingam

8. "Learning SQLite for iOS" by Gene Backlin

9. "Python and SQLite: Build Your First Database-Driven App" by Alan Beaulieu

10. Relational Database Design and Implementation: Clearly Explained by Jan L. Harrington

11. SQL For Dummies by Allen G. Taylor

12. The Definitive Guide to SQLite by Mike Owens

13. SQL Performance Explained by Markus Winand

14. Pro SQL Server Relational Database Design and Implementation by Louis Davidson, Jessica Moss, and Kevin Kline

15. High Performance MySQL: Optimization, Backups, and Replication by Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko

16. SQL Antipatterns: Avoiding the Pitfalls of Database Programming by Bill Karwin

17. The Joy of SQL: A Beginner's Guide to SQL Programming by Alan Beaulieu

18. Beginning Database Design Solutions by Rod Stephens

19. MongoDB: The Definitive Guide by Kristina Chodorow and Michael Dirolf

20. Oracle Database 12c PL/SQL Programming by Michael McLaughlin

21. NoSQL for Mere Mortals by Dan Sullivan

22. Data Modeling Essentials by Graeme Simsion and Graham Witt

23. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL by John L. Viescas

24. Database Design Manual: Using MySQL for Windows by Matthew Norman

25. Pro Oracle SQL by Karen Morton, Kerry Osborne, and Robyn Sands

26. "Python SQLite3 Tutorial" by Tutorialspoint
   - Link: https://www.tutorialspoint.com/sqlite/sqlite_python.htm
27. "Python SQLite3 Database Tutorial" by SQLite Tutorial
   - Link: https://www.sqlitetutorial.net/sqlite-python/
28. "Python SQLite3 Documentation"
   - Link: https://docs.python.org/3/library/sqlite3.html
29. "Python SQLite Tutorial" by SQLite Tutorial
   - Link: https://www.sqlitetutorial.net/sqlite-python/