Özet Tablolar:

Özellik	Mem0Tools	AGNO MemoryTools
Odak	Kullanıcı hafızası	Agent & workflow hafızası
Kalıcı/Geçici	Kalıcı	Hem kalıcı hem geçici
Semantic search	Var	Genellikle yo
Kullanıcı dışı veri	Tutmaz	Tutabilir
Entegrasyon	Mem0 API	AGNO platformu

Özellik	User Memory	Session Storage	Default Memory
Nereye kaydedilir?	tmp/memory.db	tmp/data.db	RAM (geçici)
Kimlik	user_id	session_id	Session (otomatik)
Kalıcılık	✅ Uzun vadeli (kişisel)	✅ Oturum bazlı	❌ Program kapanınca silinir
Ne saklar?	Kullanıcı bilgileri (isim, hobiler vb.)	Chat geçmişi + session state	Chat geçmişi (sadece geçici)
Kullanım senaryosu	Kişiselleştirme (ChatGPT memory gibi)	Çok turlu sohbetler	Basit kısa süreli testler

Bellek Türleri

Tür	Açıklama	İnsan Beyni Analojisi
Kısa Süreli	Session içi bağlam	Çalışma belleği
Uzun Süreli	DB'ye kaydedilen kalıcı bilgiler	Uzun vadeli hafıza
Anlamsal	Kavram, kural, semantic search	Genel bilgi
Epizodik	Zaman sıralı geçmiş olaylar	Anılar
İşlemsel	Karmaşık süreçleri otomatik çözme	Beceriler

1.Agno'da Desteklenen Bellek Biçimleri

Agno, üç temel bellek biçimini kullanıma hazır sunar:

1. **Oturum Depolama (Storage)**
    a. Chat geçmişi ve session state'i DB'de saklar.
    b. Çok turlu sohbetler ve workflow continuity için kullanılır.
2. **Kullanıcı Bellekleri (User Memories)**
    a. Kullanıcı tercihlerini uzun vadeli olarak saklar.
    b. ChatGPT'nin kişisel hafızasına benzer şekilde çalışır.
3. **Oturum Özetleri (Session Summaries)**

a. Uzun sohbetleri özetler halinde saklar.
b. Bağlamı kaybetmeden geçmişi sıkıştırır.

2.Memory (Hafıza) – Agno'daki Rolü

Bir **Agent'in hafızası**, geçmiş konuşmaları ve kullanıcıyla ilgili bilgileri hatırlayıp daha kişiselleştirilmiş yanıt vermesini sağlar.

Örneğin, kullanıcı "Ben kayak yapmayı seviyorum" derse, Agent bunu hatırlar ve ilerleyen cevaplarında buna atıf yapabilir.

Agno'da hafıza **3 ana tipte** destekleniyor:

- **Storage** → Kısa vadeli hafıza (geçmiş sohbet + state).
- **Memory** → Uzun vadeli hafıza (kullanıcı hakkında öğrenilen bilgiler).
- **Summary** → Konuşmanın özeti (bağlamı kaybetmeden geçmişi kısaltma

1. Session Storage (Oturum Depolama)

- **Ne saklar?**
  - Konuşma geçmişi (chat history)
  - Session state (örn. alışveriş listesi, sayaç, kullanıcı bilgisi vs.)
- **Özellik:**
  - Veriler veritabanına (örn. SQLite) kaydedilir.
  - Böylece oturum kapansa bile bilgiler kaybolmaz → kısa vadeli hafıza gibi çalışır.
- Agno'da buna **Storage (Depolama)** da deniyor.

2. User Memories (Kullanıcı Hafızası)

- **Ne saklar?**
  - Kullanıcıya özel öğrenilmiş bilgiler, tercihler, alışkanlıklar.
  - Örn. "John kayak yapmayı seviyor" veya "Jane sabah kahvesini seviyor".
- **Özellik:**
  - Bu bilgiler uzun vadeli saklanır.

    - Böylece Agent, bir kullanıcıyla tekrar konuştuğunda önceki tercihlerine göre cevap verebilir.
- ChatGPT'nin kişisel hafızası gibi düşünebilirsin.

3. Session Summaries (Oturum Özeti)

- **Ne saklar?**
    - Konuşmanın tamamını değil, özetini tutar.
- **Özellik:**
    - Sohbet çok uzarsa, geçmişin tamamını modele vermek pahalı/uzun olur.
    - Bunun yerine özet saklanır ve gerektiğinde Agent, bu özete bakarak bağlamı korur.

3.Memory Tools

- Session bazlı kısa süreli bilgiler (temporary memory)
- Kullanıcı veya agent geçmişi (history)
- Session özetleri, step çıktıları, workflow sonuçları

Bilgilerini tutar.

- Agent session'ı boyunca context tutar
- Database (SQLite veya başka) ile kalıcı hafıza sağlayabilir
- Kullanıcı verisi, workflow çıktısı, agent düşünceleri gibi çok yönlü bilgiler tutulabilir

- MemoryTools sınıfı, bir **ajanın (agent)** kullanıcıyla olan etkileşimlerini, geçmişini ve bağlam bilgisini saklayabilmesi için gerekli araçları sağlıyor.
- Bellek operasyonlarını (ekleme, güncelleme, silme, getirme, analiz etme) kapsayan metodlar içeriyor.
- Temelde bir **CRUD (Create, Read, Update, Delete)** yapısı ama "think" ve "analyze" gibi yapay zekaya özgü fazladan fonksiyonlarla desteklenmiş.

Normalde memory arkada sessiz çalışır.
Ama bazen ajanına **"Git hafızaya bak"** deme şansı vermek istersin.

MemoryTools bunun için var:

- Ajan → memory sorgulama, listeleme, silme işlemleri yapabilir.

- Daha kontrollü bir kullanım sağlar.

Sınıfın Yapısı

1. __init__

- Hangi araçların (tools) aktif olacağını belirliyor:
    - think, get_memories, add_memory, update_memory, delete_memory, analyze
- Bu araçları Toolkit'in içine ekliyor.
- Ayrıca başlangıçta kullanılacak talimatları (instructions) ve örnekleri (few_shot_examples) ayarlıyor.

2. think

- **Amaç:** Ajanın kendi içinde "not alması" ya da "plan yapması".
- Kullanıcıya gösterilmiyor
- session_state içinde "memory_thoughts" listesine ekleniyor.
- Örnek: "Bu kullanıcının yaş bilgisi var, ama deneyim bilgisi eksik. Önce yaşa göre filtre yapayım."

3. get_memories

- **Amaç:** Mevcut kullanıcının belleğini (önceden kaydedilmiş bilgilerini) veritabanından çekmek.
- db.get_user_memories(user_id) çağırıyor.
- Sonuçları JSON formatında döndürüyor.

4. add_memory

- **Amaç:** Yeni bir bilgi eklemek.
- Örneğin kullanıcı "Ben veganım" dedi → add_memory çağrılır.
- Yeni UserMemory nesnesi oluşturulur → DB'ye kaydedilir.
- Başarı durumunu JSON döner.

5. update_memory

- **Amaç:** Var olan bir kaydı güncellemek.
- Örneğin kullanıcı “Artık pescetaryenim” dediğinde → eski “vegan” kaydı güncellenir.
- memory_id kullanılarak DB’deki kayıt bulunur ve değiştirilir.

6. delete_memory

- **Amaç:** Gereksiz ya da güncel olmayan bir kaydı silmek.
- Örneğin kullanıcı “Eski çalışma saatlerimi unut” derse → bu fonksiyon devreye girer.

7. analyze

- **Amaç:** Yapılan bellek işlemlerinin doğru ve yeterli olup olmadığını kontrol etmek.
- “Think” gibi bu da kullanıcıya doğrudan gösterilmez, ajanın kendi iç analizi için.

8. DEFAULT_INSTRUCTIONS ve FEW_SHOT_EXAMPLES

- **DEFAULT_INSTRUCTIONS:** Bu araçların nasıl kullanılacağını açıklayan genel kurallar.
- **FEW_SHOT_EXAMPLES:** Örnek senaryolar (diyet tercihi ekleme, güncelleme, silme, hatırlatma). Bunlar ajanı eğitmek için prompt içine gömülüyor.

Memory Type	What is Stored	Human Example	Agent Example
Semantic	Facts	Things I learned in school	Facts about a user
Episodic	Experiences	Things I did	Past agent actions
Procedural	Instructions	Instincts or motor skills	Agent system prompt

4.Mem0Tools: Uzun vadeli kullanıcı hafızası

Kullanıcının verdiği bilgiler, önceki etkileşimler, agent ile konuşmaları hafızasında tutar.

- Kullanıcının verdiği bilgiler (facts, tercihler, önceki etkileşimler) hafızaya eklenir.
- Kullanıcının önceki sorguları veya agent ile yaptığı konuşmalar hatırlanabilir.
- Agent, bu bilgiler üzerinden semantik arama yapabilir veya cevaplarını buna göre özelleştirebilir.
- **Genel veriyi veya sistem bilgilerini** otomatik olarak tutmaz.
- Sadece **agent'in kullanıcıya özel bilgilerini** yönetir.
- Farklı user_id ile birden fazla kişinin hafızasını da tutabilir, ama her biri kendi kullanıcısına ait olur.

Mem0Tools Özellikleri

Mem0Tools, Mem0 API üzerinden uzun vadeli kullanıcı hafızasını yönetmek içinözel olarak geliştirilmiştir.

- **add_memory** → Kullanıcı hakkında kalıcı bilgi ekler.
- **search_memory** → Semantic search ile hafıza sorgulaması yapar.
- **get_all_memories** → Kullanıcının tüm hafızasını JSON formatında getirir.
- **delete_all_memories** → Tüm kayıtları siler.

  Kullanım senaryosu:

  "Kullanıcı kendini Fenerbahçeli olarak tanımladı" → memory'ye eklenir.

  Bir sonraki konuşmada → agent otomatik olarak bunu hatırlar.
- Workflow veya multi-step agentlarda **state management** için kullanılır.

Mem0Tools Özel Metotlar Tablosu

Metot	Görev	Parametreler	Örnek Kullanım	Çıktı
_get _user_id	Agent veya parametre üzerinden user_id çözer. Bulunamazsa hata verir.	agent, user_id (opsiyonel)	mem_tool._get_user_id(agent)	user_123 veya hata mesajı
add_memory	Kullanıcıya ait yeni bilgi/fact ekler.	content (str veya dict olabilir)	mem_tool.add_memory(agent, "I live in NYC")	Ekleme işlemi sonucu (JSON)

search_memory	**Semantic search ile hafızada bilgi arar.**	**query (str)**	**mem_tool.search_memory(agent, "Where does John live?")**	**İlgili memory listesi**
get_all_memories	Kullanıcının tüm hafızasını JSON formatında döner.	agent, user_id (opsiyonel)	mem_tool.get_all_memories(agent)	JSON memory listesi
delete_all_memories	Kullanıcının hafızasındaki tüm kayıtları siler.	agent, user_id (opsiyonel)	mem_tool.delete_all_memories(agent)	Silme işlemi sonucu

5.Memory Flags ve Otomatik Hatırlama

Belleğin ne kadar ve nasıl kullanılacağını belirleyen bazı bayraklar vardır:

- **add_history_to_messages** → Son n konuşmayı otomatik olarak her mesaja ekler.
- **read_chat_history** → Tüm sohbet geçmişini okuma aracı ekler.
- **read_tool_call_history** → Geçmişte yapılan tool çağrılarını görme imkânı verir.

Bu bayraklar sayesinde agent, ne kadar geçmişi bağlama dahil edeceğini otomatik kontrol edebilir.

Default Memory (Varsayılan Bellek)

Her Agent'in, içinde bulunduğu **oturumdaki konuşmaları (chat history)** hatırlama özelliği vardır.

- **read_tool_call_history=True**
  - Bu ayarı açarsan, Agent'e bir **get_tool_call_history() aracı** eklenir.

- Bu araç sayesinde Agent, geçmişte yapılan **tool çağrılarını (örneğin API çağrıları)** ters kronolojik sırayla (en son yapılan en başta olacak şekilde) okuyabilir.

Kullanıcı Hafızası (User Memory) → *Uzun vadeli kişiselleştirilmiş bellek*

```
memory_db = SqliteMemoryDb(table_name="memory", db_file="tmp/memory.db")
memory = Memory(db=memory_db)

agent = Agent(
   model=Gemini(id="gemini-2.0-flash-exp"),
   memory=memory,
   enable_agentic_memory=True,
)
```

Amaç: kullanıcı hakkında uzun vadeli bilgiler saklamak.

User_id ile ilişkilendirilir, tmp/memory.db'de kalıcı olarak tutulur.

Session Storage (Chat History + Session State) → *Kısa vadeli oturum belleği*

```
agent = Agent(
   model=OpenAIChat(id="gpt-4o-mini"),
   session_id="fixed_id_for_demo",
   storage=SqliteStorage(table_name="agent_sessions", db_file="tmp/data.db"),
   add_history_to_messages=True,
   num_history_runs=3,
)
```

- **Amaç:**

Sadece o **oturumda konuşulanları** saklamak.

- **Nasıl çalışır:**
    - session_id="fixed_id_for_demo" ile aynı oturuma ait mesajlar bir araya getirilir.
    - tmp/data.db içinde saklanır.
    - Son n mesaj modelin bağlamına otomatik eklenebilir (num_history_runs=3).
- **Özellik:**
    - Sen “What was my last question?” dediğinde, bunu sadece bu session DB’sinden bulur.
    - Kişisel bilgi saklamaz. Yani kullanıcı adı ya da hobileri uzun vadeli hatırlamaz.
- **Kullanım senaryosu:**

Chatbot’un bir konuşma sırasında **bağlamı kaybetmemesi** için.

Default Memory (Built-in Chat History) → *Anlık bellek (Agent içinde)*

```
agent = Agent(
   model=Gemini(id="gemini-2.0-flash-exp"),
   add_history_to_messages=True,
   num_history_responses=3,
)
```

- **Amaç:**

Agent kendi içinde **geçici** olarak mesaj geçmişini tutuyor.

- **Nasıl çalışır:**
    - DB’ye kaydetmez.
    - Hafıza sadece RAM’de, o anki çalıştırmada mevcut.
    - get_messages_for_session() çağrısı ile anlık mesaj geçmişini alabiliyorsun.
- **Özellik:**
    - Program kapanınca tüm geçmiş kaybolur.
    - num_history_responses=3 diyerek son 3 mesajı her isteğe ekler.
- **Kullanım senaryosu:**

Hızlı testler, **DB kullanmadan geçici bellek**.

6.Memory Manager

Görevi

- Kullanıcı mesajlarından önemli bilgileri **yakalar** (capture).
- DB'ye kaydeder (SQLite, Postgres, MongoDB).
- Daha sonra arama, güncelleme, silme işlemlerine açar.
- Bir nevi "memory polisi" gibi davranır: hangi bilginin tutulup tutulmayacağına karar verir.

- **Bir agent gibi davranır**:
    - Hafızası vardır.
    - Tools kullanır (add, update, delete, clear memory).
    - LLM ile konuşarak karar verir

Fonksiyonlar

- create_user_memories → Yeni memory oluştur / güncelle
- replace_user_memory → Var olanı yenisiyle değiştir
- update_memory_task → Görev cümlesine göre memory güncelle
- search_user_memories → Semantic arama (last_n, first_n, agentic)
- get_user_memories → Bir user_id'nin tüm memory'lerini getir
- get_user_memory → Tek memory çek
- clear_memories → Kullanıcıya ait tüm memoryleri sil

Ana Bileşenler

1. **Model (self.model)**
    a. MemoryManager, arka planda bir LLM kullanır (örn. OpenAIChat).
    b. LLM'e "hangi memory güncellensin / eklensin?" diye danışır.
2. **System Message**
    a. LLM'e verilen görev tanımıdır.
    b. İçinde:
        i. "Ne zaman memory ekle/güncelle/sil" kuralları
        ii. Kullanıcının mevcut memory'leri
        iii. Capture kriterleri (örn. isim, tercih, geçmiş olaylar)

    c. Böylece model “memory polisi” gibi davranır.
3. **Tools (Fonksiyonlar)**
    a. add_memory, update_memory, delete_memory, clear_memory fonksiyonları.
    b. Bunlar LLM’in çağırabileceği **araçlar**dır.
    c. determine_tools_for_model ile modele tanıtılır.
4. **DB Katmanı (BaseDb)**
    a. Memoryler burada saklanır.
    b. SQLite veya Postgres kullanılabilir.
    c. read_from_db, upsert_user_memory, delete_user_memory, clear_memories metodları üzerinden çalışır.

Önemli Fonksiyonlar

create_user_memories

- Kullanıcıdan gelen mesaj(lar)ı alır.
- Mevcut memory’lerle karşılaştırır.
- Modeli çalıştırır → memory ekle/güncelle kararını verir.
- Sonucu DB’ye yazar.

update_memory_task

- Dışarıdan verilen bir “task” stringini (örn. “Kullanıcının adı artık Ayşe değil Elif”) modele yollar.
- Model, uygun aracı çağırarak memory’yi günceller.

search_user_memories

- Kullanıcının memory’leri arasında arama yapar.
- Yöntemler:
    - last_n: En son eklenenler
    - first_n: En eski memory’ler
    - agentic: LLM tabanlı semantic search

_get_last_n_memories & _get_first_n_memories

- Memory’leri tarih sırasına göre sıralar.

- datetime.min ve datetime.max default değerler olarak kullanılır (timestamp boşsa).

_search_user_memories_agentic

- Memory'leri bir liste halinde modele gönderir.
- "Şu query ile alakalı olan memory ID'lerini döndür" der.
- Model, ID listesi döndürür → sistem ilgili memory'leri seçer.

- MemoryManager: Hafızayı yöneten sınıf.
- Kendi ayarlarını yapabilirsin:
  - Hangi modeli kullansın (ör. gpt-5-mini).
  - Ek talimat verebilirsin (örn. "kullanıcının gerçek adını kaydetme").

Örnek:

```
memory_manager = MemoryManager(
  db=db,
  model=OpenAIChat(id="gpt-5-mini"),
  additional_instructions="Don't store the user's real name",
)
```

Böylece kullanıcı "Benim adım Ahmet" derse, hafıza **"Kullanıcı kendini Ahmet olarak tanıtıyor"** diye kaydeder ama gerçek adı olarak yazmaz.

Context ile Memory farkı

- **add_memories_to_context=True (default)**
  → Hafıza bilgileri her cevaba otomatik olarak eklenir.
  (Kullanıcı sorunca "Ha evet senin hobin futbol" diyebilir.)
- **add_memories_to_context=False**
  → Hafıza tutulur ama her mesaja sokulmaz.
  (Sen özel olarak sorarsan veya memory tools ile çekersen gösterir.)

Mem0Tools vs Agno MemoryTools

Özellik	Mem0Tools	Agno MemoryTools
Odak	Kullanıcı hafızası	Agent & workflow hafızası
Kalıcılık	Kalıcı	Hem kalıcı hem geçici
Semantic Search	Var	Genellikle yok
Kullanıcı Dışı Veri	Tutmaz	Tutabilir
Entegrasyon	Mem0 API	Agno platformu