

backend\app\memory\models.py

```
1  from datetime import datetime
2  from sqlalchemy import (
3      Column, Integer, String, Text, ForeignKey, DateTime, Boolean, Index, JSON
4  )
5  from sqlalchemy.orm import declarative_base, relationship
6
7  Base = declarative_base()
8
9  __all__ = [
10     "Base",
11     "Role",
12     "Project",
13     "RoleProjectLink",
14     "MemoryEntry",
15     "CanonItem",
16     "PromptTemplate",
17     "AuditLog",
18     "Attachment",
19 ]
20
21 # ✅ Role table
22 class Role(Base):
23     __tablename__ = "roles"
24
25     id = Column(Integer, primary_key=True)
26     name = Column(String(50), unique=True, nullable=False, index=True)
27     description = Column(String(255), nullable=True)
28
29     # Relationships (no cascading deletes to MemoryEntry/CanonItem; DB handles SET NULL)
30     memories = relationship(
31         "MemoryEntry",
32         back_populates="role",
33         passive_deletes=True,
34     )
35     role_projects = relationship("RoleProjectLink", back_populates="role", cascade="all, delete")
36     prompt_templates = relationship("PromptTemplate", back_populates="role", cascade="all, delete")
```

```


37     canon_items = relationship(
38         "CanonItem",
39         back_populates="role",
40         passive_deletes=True,
41     )
42
43     def __repr__(self) -> str:
44         return f"<Role id={self.id} name={self.name!r}>"
45
46 # ✅ Project table
47 class Project(Base):
48     __tablename__ = "projects"
49
50     id = Column(Integer, primary_key=True)
51     name = Column(String(100), unique=True, nullable=False, index=True)
52     description = Column(Text, nullable=True)
53     project_structure = Column(Text, nullable=True)
54
55     # View-only mirrors for FK-int join convenience
56     memories = relationship(
57         "MemoryEntry",
58         back_populates="project",
59         primaryjoin="foreign(MemoryEntry.project_id_int)==Project.id",
60         viewonly=True,
61     )
62     role_projects = relationship("RoleProjectLink", back_populates="project", cascade="all, delete")
63     canon_items = relationship(
64         "CanonItem",
65         back_populates="project",
66         primaryjoin="foreign(CanonItem.project_id_int)==Project.id",
67         viewonly=True,
68     )
69
70     def __repr__(self) -> str:
71         return f"<Project id={self.id} name={self.name!r}>"
72
73 # ✅ Link table for Role ↔ Project (many-to-many)
74 class RoleProjectLink(Base):

```

```

75     __tablename__ = "role_project_link"
76
77     id = Column(Integer, primary_key=True)
78     role_id = Column(Integer, ForeignKey("roles.id", ondelete="CASCADE"), nullable=False)
79     project_id = Column(Integer, ForeignKey("projects.id", ondelete="CASCADE"), nullable=False)
80
81     role = relationship("Role", back_populates="role_projects")
82     project = relationship("Project", back_populates="role_projects")
83
84     __table_args__ = (
85         Index("ix_role_project_unique", "role_id", "project_id", unique=True),
86     )
87
88     def __repr__(self) -> str:
89         return f"<RoleProjectLink role_id={self.role_id} project_id={self.project_id}>"
90
91 # ☒ Memory entries (chat rows + summaries)
92 class MemoryEntry(Base):
93     __tablename__ = "memory_entries"
94
95     id = Column(Integer, primary_key=True)
96
97     # Dual key: string for API filtering, int for FK joins/analytics
98     project_id = Column(String(255), nullable=False, index=True)
99     project_id_int = Column(Integer, ForeignKey("projects.id", ondelete="CASCADE"), nullable=True)
100
101     # NOTE: ON DELETE SET NULL aligns with passive_deletes on Role.memories
102     role_id = Column(Integer, ForeignKey("roles.id", ondelete="SET NULL"), nullable=True, index=True)
103     chat_session_id = Column(String(255), nullable=True, index=True)
104
105     timestamp = Column(DateTime, default=datetime.utcnow, index=True)
106     tokens = Column(Integer, nullable=True)
107     summary = Column(Text, nullable=True)
108     raw_text = Column(Text, nullable=True)
109
110     # Flags
111     is_summary = Column(Boolean, default=False, nullable=False, index=True)
112     is_ai_to_ai = Column(Boolean, default=False, nullable=False)

```

```
113
114     deleted = Column(Boolean, default=False, nullable=False, index=True)
115     updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
116
117     role = relationship(
118         "Role",
119         back_populates="memories",
120         passive_deletes=True,
121     )
122     project = relationship(
123         "Project",
124         back_populates="memories",
125         primaryjoin="foreign(MemoryEntry.project_id_int)==Project.id",
126         viewonly=True,
127     )
128
129     __table_args__ = (
130         Index("ix_mem_role_proj_sess_time", "role_id", "project_id", "chat_session_id", "timestamp"),
131         Index("ix_mem_proj_role_is_summary_time", "project_id", "role_id", "is_summary", "timestamp"),
132     )
133
134     attachments = relationship(
135         "Attachment",
136         back_populates="message",
137         cascade="all, delete-orphan",
138         passive_deletes=True,
139     )
140
141     def __repr__(self) -> str:
142         return f"<MemoryEntry id={self.id} role_id={self.role_id} proj={self.project_id} sess={self.chat_session_id}>"
143
144 #  Canonical items (ADR, CHANGELOG, BACKLOG, GLOSSARY, PMD)
145 class CanonItem(Base):
146     __tablename__ = "canon_items"
147
148     id = Column(Integer, primary_key=True)
149
150     project_id = Column(String(255), nullable=False, index=True)
```

```

151     project_id_int = Column(Integer, ForeignKey("projects.id", ondelete="CASCADE"), nullable=True)
152
153     role_id = Column(Integer, ForeignKey("roles.id", ondelete="SET NULL"), nullable=True, index=True)
154
155     type = Column(String(32), nullable=False, index=True)
156     title = Column(String(256), nullable=False, index=True)
157     body = Column(Text, nullable=False)
158
159     tags = Column(JSON, nullable=True) # SQLite stores as TEXT; SQLAlchemy handles it
160     terms = Column(Text, nullable=True)
161
162     created_at = Column(DateTime, default=datetime.utcnow, index=True)
163     updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
164     is_active = Column(Boolean, default=True, nullable=False, index=True)
165
166     role = relationship(
167         "Role",
168         back_populates="canon_items",
169         passive_deletes=True,
170     )
171     project = relationship(
172         "Project",
173         back_populates="canon_items",
174         primaryjoin="foreign(CanonItem.project_id_int)==Project.id",
175         viewonly=True,
176     )
177
178     __table_args__ = (
179         Index("ix_canon_proj_role_type_time", "project_id", "role_id", "type", "created_at"),
180         Index("ix_canon_title_terms", "title", "terms"),
181     )
182
183     def __repr__(self) -> str:
184         return f"<CanonItem id={self.id} type={self.type} title={self.title!r}>"
185
186 # ✅ Prompt templates tied to a role
187 class PromptTemplate(Base):
188     __tablename__ = "prompt_templates"

```

```

189
190     id = Column(Integer, primary_key=True)
191     role_id = Column(Integer, ForeignKey("roles.id", ondelete="CASCADE"), nullable=False, index=True)
192     name = Column(String(100), nullable=False)
193     content = Column(Text, nullable=False)
194     is_default = Column(Boolean, default=False)
195
196     role = relationship("Role", back_populates="prompt_templates")
197
198     __table_args__ = (
199         Index("ix_prompt_unique_per_role", "role_id", "name", unique=True),
200     )
201
202     def __repr__(self) -> str:
203         return f"<PromptTemplate id={self.id} role_id={self.role_id} name={self.name!r}>"
204
205 # ✅ Audit log entries with model version
206 class AuditLog(Base):
207     __tablename__ = "audit_logs"
208
209     id = Column(Integer, primary_key=True)
210
211     project_id = Column(String(255), nullable=False, index=True)
212     role_id = Column(Integer, nullable=False, index=True)
213     chat_session_id = Column(String(255), nullable=True, index=True)
214
215     provider = Column(String(50), nullable=False, index=True)
216     action = Column(String(50), nullable=False)
217     query = Column(Text, nullable=True)
218     timestamp = Column(DateTime, default=datetime.utcnow, index=True)
219     model_version = Column(String(50), nullable=True)
220
221     def __repr__(self) -> str:
222         return f"<AuditLog id={self.id} provider={self.provider} action={self.action}>"
223
224 # ✅ Attachments for file uploads
225 class Attachment(Base):
226     __tablename__ = "attachments"

```

```
227
228     id = Column(Integer, primary_key=True, index=True)
229
230     # Link to message
231     message_id = Column(Integer, ForeignKey("memory_entries.id", ondelete="CASCADE"), nullable=False, index=True)
232
233     # File info
234     filename = Column(String(255), nullable=False)
235     original_filename = Column(String(255), nullable=False)
236     file_type = Column(String(50), nullable=False) # image, document, data
237     mime_type = Column(String(100), nullable=False)
238     file_size = Column(Integer, nullable=False)
239
240     # File path
241     file_path = Column(String(500), nullable=False)
242
243     # Metadata
244     uploaded_at = Column(DateTime, default=datetime.utcnow, nullable=False)
245
246     # Relations
247     message = relationship(
248         "MemoryEntry",
249         back_populates="attachments",
250         passive_deletes=True,
251     )
252
253     def __repr__(self) -> str:
254         return f"<Attachment(id={self.id}, filename={self.filename}, type={self.file_type})>"
```